

バージョン 2.x のデベロッパーガイド

# AWS SDK for Java 2.x



# AWS SDK for Java 2.x: バージョン 2.x のデベロッパーガイド

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

Amazon の商標とトレードドレスは、Amazon 以外の製品またはサービスとの関連において、顧客に混乱を招いたり、Amazon の名誉または信用を毀損するような方法で使用することはできません。Amazon が所有しない他の商標はすべてそれぞれの所有者に帰属します。所有者は必ずしも Amazon との提携や関連があるわけではありません。また、Amazon の支援を受けているとはかぎりません。

# Table of Contents

デベロッパーガイド - AWS SDK for Java 2.x .....	1
SDK の使用を開始する .....	1
モバイルアプリケーションの開発 .....	1
SDK メジャーバージョンのメンテナンスとサポート .....	1
追加リソース .....	2
この SDK に貢献する .....	2
開始方法のチュートリアル .....	3
ステップ 1: このチュートリアルのために設定する .....	3
ステップ 2: プロジェクトを作成する .....	3
ステップ 3: コードを記述する .....	8
ステップ 4: アプリケーションを構築して実行する .....	12
成功 .....	13
クリーンアップ .....	13
次のステップ .....	14
セットアップ .....	15
設定の概要 .....	15
認証を設定する。 .....	16
SDK のシングルサインオンアクセスのセットアップ .....	16
を使用してサインインする AWS CLI .....	17
Java と構築ツールをインストールする .....	18
追加認証オプション .....	18
Apache Maven プロジェクトを設定する .....	18
前提条件 .....	18
Maven プロジェクトを作成する .....	19
Maven 用の Java コンパイラを設定する .....	20
SDK を依存関係として宣言する .....	21
SDK モジュールの依存関係を設定する .....	22
プロジェクトを構築する .....	24
Gradle プロジェクトを設定する .....	25
GraalVM Native Image プロジェクトを設定する .....	31
前提条件 .....	31
アーキタイプを使用してプロジェクトを作成する .....	32
ネイティブイメージを構築する .....	33
SDK を使用する .....	34

サービスクライアントを使用する .....	34
サービスクライアントを作成する .....	34
デフォルトのクライアント設定 .....	35
サービスクライアントを設定する .....	35
リクエストを発行する .....	36
レスポンスを処理する .....	36
サービスクライアントを閉じる .....	37
例外を処理する .....	38
ウェイターを使用する .....	39
HTTP クライアント .....	39
再試行 .....	40
タイムアウト .....	40
実行インターセプター .....	41
追加情報 .....	41
SDK への一時的な認証情報の提供 .....	41
一時的な認証情報へのアクセスを設定する .....	42
デフォルトの認証情報プロバイダーチェーン .....	43
特定の認証情報プロバイダーまたはプロバイダーチェーンを使用する .....	45
プロファイルを使用する .....	46
外部プロセスからの一時的な認証情報をロードする .....	49
コードに一時的な認証情報を入力する .....	52
での IAM ロール認証情報の読み取り Amazon EC2 .....	55
の使用 AWS リージョン .....	57
AWS リージョンを明示的に設定する .....	57
環境からリージョンを決定する .....	58
サービス可用性を確認する .....	59
特定のエンドポイントを選択する .....	60
の SDK の起動時間を短縮する AWS Lambda .....	60
AWS CRT ベースの HTTP クライアントを使用する .....	60
未使用の HTTP クライアント依存関係を削除する .....	61
検索をショートカットするようにサービスクライアントを設定します。 .....	62
Lambda 関数ハンドラーの外部で SDK クライアントを初期化する .....	63
依存関係インジェクションを最小限に抑える .....	63
Maven アーケタイプターゲットを使用する AWS Lambda .....	64
Lambda SnapStart .....	64
起動時間に影響するバージョン 2.x の変更点 .....	64

追加リソース .....	64
HTTP クライアント .....	65
利用可能なクライアント .....	65
クライアントの推奨事項 .....	66
スマートのデフォルト .....	69
Apache ベースの HTTP クライアントの設定 .....	71
URLConnection ベースの HTTP クライアントを設定する .....	76
Netty ベースの HTTP クライアントの設定 .....	83
AWS CRT ベースの HTTP クライアントを設定する .....	89
HTTP プロキシの設定 .....	100
例外処理 .....	105
非チェック例外を使用する理由 .....	106
AwsServiceException ( およびサブクラス ) .....	106
SdkClientException .....	107
例外と再試行動作 .....	107
ログ記録 .....	107
Log4j 2 設定ファイル .....	108
ログ記録を追加する .....	108
SDK 固有のエラーおよび警告 .....	109
リクエスト/応答の概要のログ記録 .....	110
デバッグレベルの SDK ログ記録 .....	111
ワイヤログ記録を有効にする .....	113
DNS 名参照用の JVM TTL を設定する .....	118
JVM TTL を設定する方法 .....	119
ベストプラクティス .....	119
SDK クライアントを再利用する .....	120
入カストリームを閉じる .....	120
HTTP 設定を調整する .....	120
Netty に OpenSSL を使用 .....	120
API タイムアウトの設定 .....	121
メトリクスを使用 .....	122
トラブルシューティング .....	122
接続のリセット .....	122
接続タイムアウト .....	123
読み取りタイムアウト .....	124
接続プールのタイムアウト .....	124

クラスパスエラー .....	128
署名エラー .....	129
接続プールのシャットダウン .....	130
SDK 機能を使用する .....	133
一般的な機能 .....	133
サービス固有の機能 .....	133
ページ分割された結果を処理する .....	133
同期ページ割り .....	134
非同期ページ割り .....	137
リソースの状態をポーリングする .....	142
前提条件 .....	142
ウェーターの使用 .....	143
ウェーターを設定する .....	144
コードの例 .....	145
非同期プログラミングを使用する .....	145
非ストリーミング操作 .....	145
ストリーミング操作 .....	148
高度な操作 .....	152
HTTP/2 を使用する .....	153
SDK メトリクスを使用する .....	154
前提条件 .....	154
メトリクスの収集を有効にする方法 .....	155
メトリクスパブリッシャーをカスタマイズする .....	156
メトリクスが利用可能になるまでの時間 .....	158
収集される情報 .....	158
この情報の使用方法 .....	159
サービスクライアントのメトリクス .....	159
の操作 AWS のサービス .....	164
CloudWatch .....	164
CloudWatch からメトリクスを取得 .....	165
カスタムメトリクスデータを CloudWatch にパブリッシュ .....	167
CloudWatch アラームを操作する .....	169
Amazon CloudWatch Events を使用する .....	173
AWS データベースサービス .....	176
Amazon DynamoDB .....	177
Amazon RDS .....	178

Amazon Redshift .....	178
Amazon Aurora Serverless v1 .....	178
Amazon DocumentDB .....	179
DynamoDB .....	179
でのテーブルの操作 DynamoDB .....	179
DynamoDB の項目の操作 .....	189
DynamoDB アイテムにオブジェクトをマッピングする .....	196
Amazon EC2 .....	314
Amazon EC2 インスタンスを管理する .....	315
AWS リージョン とアベイラビリティゾーンの使用 .....	321
でセキュリティグループを操作する Amazon EC2 .....	325
Amazon EC2 インスタンスメタデータを操作する .....	330
IAM .....	336
IAM アクセスキーの管理 .....	336
IAM ユーザーを管理する .....	342
IAM ポリシーを作成する .....	347
IAM ポリシーの使用 .....	354
IAM サーバー証明書の使用 .....	361
Kinesis .....	365
Amazon Kinesis Data Streams へのサブスクライブ .....	366
Lambda .....	376
Lambda 関数を呼び出します。 .....	377
Lambda 関数を一覧表示する .....	378
Lambda 関数を削除する .....	379
Amazon S3 .....	380
アクセスポイントまたはマルチリージョンアクセスポイントを使用する .....	380
バケットオペレーション .....	382
オブジェクト操作 .....	389
署名付き URL .....	399
クロスリージョンアクセス .....	407
チェックサム .....	409
パフォーマンスの高い S3 クライアントを使用する .....	410
ファイルとディレクトリを転送する .....	413
Amazon SNS .....	421
トピックの作成 .....	421
Amazon SNS トピックをリストする .....	423

トピックへのエンドポイントのサブスクライブ .....	423
トピックへのメッセージの発行 .....	425
トピックからのエンドポイントの登録解除 .....	426
トピックの削除 .....	427
Amazon SQS .....	428
キューオペレーション .....	428
メッセージオペレーション .....	431
Amazon Transcribe .....	435
マイクを設定する .....	435
パブリッシャーを作成する .....	436
クライアントを作成してストリーミングを開始する .....	438
詳細情報 .....	434
コードの例 .....	441
一般的なシナリオのシナリオ .....	441
API Gateway .....	443
Application Auto Scaling .....	447
Application Recovery Controller .....	456
Aurora .....	459
Auto Scaling .....	494
Amazon Bedrock .....	556
Amazon Bedrock ランタイム .....	562
CloudFront .....	642
CloudWatch .....	662
CloudWatch イベント .....	713
CloudWatch ログ .....	719
Amazon Cognito ID .....	730
Amazon Cognito Identity Provider .....	737
Amazon Comprehend .....	765
DynamoDB .....	776
Amazon EC2 .....	855
Amazon ECS .....	927
Elastic Load Balancing - バージョン 2 .....	941
MediaStore .....	986
OpenSearch サービス .....	1002
EventBridge .....	1010
Forecast .....	1042



AWS Glue .....	1056
HealthImaging .....	1080
IAM .....	1108
AWS IoT .....	1193
AWS IoT data .....	1220
Amazon Keyspaces .....	1222
Kinesis .....	1249
AWS KMS .....	1262
Lambda .....	1299
MediaConvert .....	1324
Migration Hub .....	1347
Amazon Personalize .....	1360
Amazon Personalize Events .....	1391
Amazon Personalize Runtime .....	1394
Amazon Pinpoint .....	1399
Amazon Pinpoint SMS および音声 API .....	1444
Amazon Polly .....	1448
Amazon RDS .....	1454
Amazon Redshift .....	1495
Amazon Rekognition .....	1521
Route 53 ドメイン登録 .....	1589
Amazon S3 .....	1612
S3 Glacier .....	1753
SageMaker .....	1770
Secrets Manager .....	1799
Amazon SES .....	1801
Amazon SES API v2 .....	1814
Amazon SNS .....	1832
Amazon SQS .....	1881
Step Functions .....	1901
AWS STS .....	1925
AWS Support .....	1929
Systems Manager .....	1952
Amazon Textract .....	1982
Amazon Transcribe .....	1993
クロスサービスの例 .....	2010

DynamoDB テーブルにデータを送信するアプリケーションを構築する .....	2010
Amazon Lex chatbot を構築する .....	2011
Amazon SNS アプリケーションの構築 .....	2011
メッセージングアプリケーションを作成する .....	2012
サーバーレスアプリケーションを作成して写真の管理 .....	2012
DynamoDB データを追跡するウェブアプリケーションを作成する .....	2013
Amazon Redshift データ追跡用のウェブアプリケーションの作成 .....	2013
Aurora Serverless 作業項目トラッカーの作成 .....	2013
顧客からのフィードバックを分析するアプリケーションの作成 .....	2014
イメージ内の PPE を検出する .....	2015
イメージ内のオブジェクトを検出する .....	2015
動画内の人物や物体を検出する .....	2016
DynamoDB のパフォーマンスのモニタリング .....	2016
メッセージをキューに発行する .....	2017
API Gateway を使用して Lambda 関数を呼び出す .....	2017
Step Functions を使用して Lambda 関数を呼び出す .....	2018
スケジュールされたイベントを使用した Lambda 関数の呼び出し .....	2018
セキュリティ .....	2019
データ保護 .....	2019
Transport Layer Security (TLS) .....	2020
TLS バージョンの確認 .....	2021
TLS バージョンの強制 .....	2021
TLS 1.2 に移行する .....	2022
Identity and Access Management .....	2022
対象者 .....	2022
アイデンティティを使用した認証 .....	2023
ポリシーを使用したアクセスの管理 .....	2027
IAM の AWS のサービス 仕組み .....	2029
AWS ID とアクセスのトラブルシューティング .....	2029
コンプライアンス検証 .....	2031
耐障害性 .....	2033
インフラストラクチャセキュリティ .....	2033
バージョン 2 に移行する .....	2035
バージョン 2 の新機能 .....	2035
S tep-by-step 命令 .....	2036
手順の概要 .....	2036

移行例 .....	2037
パッケージ名から artifactId へのマッピング .....	2048
命名規則 .....	2049
例外 .....	2049
1.x と 2.x の相違点 .....	2054
パッケージ名の変更 .....	2054
プロジェクトへのバージョン 2.x の追加 .....	2055
イミュータブルな POJO .....	2055
セッターメソッドとゲッターメソッド .....	2056
モデルクラス名 .....	2056
ライブラリとユーティリティ .....	2057
クライアントの変更 .....	2059
認証情報プロバイダーの変更 .....	2103
リージョンの変更 .....	2111
オペレーション、リクエスト、レスポンスの変更 .....	2113
例外変更 .....	2114
シリアル化の変更 .....	2116
サービス固有の変更 .....	2117
プロファイルファイルの変更 .....	2122
外部設定 .....	2124
ウェーター .....	2127
S3 Transfer Manager .....	2130
EC2 メタデータユーティリティ .....	2137
CloudFront 事前署名 .....	2144
S3 URI 解析 .....	2148
IAM Policy Builder API .....	2150
SDK for Java 1.x と 2.x を並行して使用する .....	2156
OpenPGP 鍵 .....	2158
現在のキー .....	2158
ドキュメント履歴 .....	2160
.....	mmclxviii

# デベロッパーガイド - AWS SDK for Java 2.x

AWS SDK for Java には、AWS のサービス用の Java API が用意されています。この SDK を使用すると、Amazon S3、Amazon EC2、DynamoDB など動作する Java アプリケーションを構築できます。

AWS SDK for Java 2.x は、1.x コードベースより大幅に変更されています。本バージョンは Java 8 以降に構築されており、要望の声が高い機能が複数追加されています。このバージョンでは、ノンブロッキング I/O がサポートされており、ランタイム時にさまざまな HTTP 実装をプラグインすることができます。

AWS SDK for Java には、新しいサービスのサポートが定期的に追加されています。特定のバージョンにおける変更点や機能の一覧は、「[変更ログ](#)」を表示してください。

## SDK の使用を開始する

SDK を実際に使用する準備ができている場合は、[開始方法のチュートリアル](#) チュートリアルに従ってください。

開発環境を設定するには、「[セットアップ](#)」を参照してください。

現在 SDK for Java のバージョン 1.x を使用している場合、具体的なガイダンスについては、「[Migrate to version 2](#)」を参照してください。

Amazon S3、DynamoDB、Amazon EC2、およびその他の AWS のサービスにリクエストを行う方法については、「[Use the SDK for Java](#)」および「[Work with AWS のサービス](#)」を参照してください。

## モバイルアプリケーションの開発

お客様がモバイルアプリ開発者の場合、Amazon Web Services では [AWS Amplify](#) フレームワークを提供します。

## SDK メジャーバージョンのメンテナンスとサポート

SDK メジャーバージョンのメンテナンスとサポート、およびその基礎的な依存関係については、「[AWS SDKs and Tools リファレンスガイド](#)」で以下を参照してください。

- [AWS SDK とツールのメンテナンスポリシー](#)
- [AWS SDK とツールのバージョンサポートマトリクス](#)

## 追加リソース

このガイドに加えて、以下の AWS SDK for Java 開発者のための貴重なオンラインリソースもあります。

- [AWS SDK for Java 2.x API リファレンス](#)
- [Java 開発者ブログ](#)
- [AWS re:Post の Java 開発トピック](#)
- GitHub 上の [SDK ソース](#)
- [AWS SDK コードサンプルライブラリ](#)
- [@awsforjava \(Twitter\)](#)

## この SDK に貢献する

また、開発者は、次のチャンネルを使用してフィードバックを送信することができます。

- GitHub で問題を送信:
  - [デベロッパーガイドのドキュメントに関する問題を送信](#)
  - [SDK に関する問題を送信](#)
- AWS SDK for Java 2.x [Gitter チャンネル](#)でこの SDK に関する非公式チャットに参加する

# AWS SDK for Java 2.x の使用を開始する

AWS SDK for Java 2.x には、Amazon Web Services ( ) 用の Java APIs が用意されています。AWS SDK を使用すると、Amazon S3、Amazon EC2、DynamoDBなどで動作する Java アプリケーションを構築できます。

このチュートリアルでは、[Apache Maven](#) を使用して SDK for Java 2.x の依存関係を定義し、Amazon S3 に接続してファイルをアップロードするコードを記述する方法を説明します。

このチュートリアルを完了するには、次の手順に従ってください。

- [ステップ 1: このチュートリアルのために設定する](#)
- [ステップ 2: プロジェクトを作成する](#)
- [ステップ 3: コードを記述する](#)
- [ステップ 4: アプリケーションを構築して実行する](#)

## ステップ 1: このチュートリアルのために設定する

このチュートリアルを開始する前に、以下を実行する必要があります

- アクセス許可 Amazon S3
- へのシングルサインオン AWS のサービス を使用して にアクセスするように設定された Java 開発環境 AWS IAM Identity Center

[???](#) の手順に従って、このチュートリアルの設定を行ってください。Java SDK の [シングルサインオンアクセスを使用して開発環境を設定し、アクティブな AWS アクセスポータルセッション](#) を作成したら、このチュートリアルのステップ 2 に進みます。

## ステップ 2: プロジェクトを作成する

このチュートリアル用のプロジェクトを作成するには、プロジェクトの設定方法に関する入力を求める Maven コマンドを実行します。すべての入力と確認が完了すると、Maven は pom.xml を作成してプロジェクトの構築を終了し、スタブ Java ファイルを作成します。

1. ターミナルまたはコマンドプロンプトウィンドウを開き、Desktop や Home フォルダなど、任意のディレクトリに移動します。

2. ターミナルに以下のコマンドを入力して、Enter を押します。

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

3. 各プロンプトの 2 列目にリストされている値を入力します。

プロンプト	入力する値
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. 最後の値を入力すると、Maven は選択した内容を一覧表示します。Y を入力して確認するか、N を入力して値を再入力します。

Maven は、入力した `getstarted` 値に基づいて `artifactId` という名前が付けられたプロジェクトフォルダーを作成します。getstarted フォルダー内で、レビューできる `README.md` ファイル、`pom.xml` ファイル、および `src` ディレクトリを検索します。

Maven は以下のディレクトリツリーを構築します。

```
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

以下は、`pom.xml` プロジェクトファイルの内容を示しています。

## `pom.xml`

`dependencyManagement` セクションには AWS SDK for Java 2.x に対する依存関係が含まれており、`dependencies` セクションには Amazon S3 に対する依存関係があります。プロジェクトでは Java 1.8 が使用されています。これは、`maven.compiler.source`、`maven.compiler.target` およびプロパティに 1.8 値があるためです。

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



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

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

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

```
<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>
```

## ステップ 3: コードを記述する

次のコードは Maven によって作成された App クラスを示しています。main メソッドはアプリケーションへのエントリーポイントであり、Handler クラスのインスタンスを作成してその sendRequest メソッドを呼び出します。

### App クラス

```
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 が作成する `DependencyFactory` クラスには、[S3Client](#) インスタンスを構築して返す `s3Client` ファクトリメソッドが含まれています。S3Client インスタンスは Apache ベースの HTTP クライアントのインスタンスを使用します。これは、どの HTTP クライアントを使用するかを Maven が求めたときに `apache-client` が指定されたためです。

`DependencyFactory` は次のコードに示されています。

## DependencyFactory クラス

```
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` クラスにはプログラムのメインロジックが含まれています。`Handler` のインスタンスが `App` クラス内で作成されると、`DependencyFactory` は `S3Client` サービスクライアントを配置します。コードは、`S3Client` インスタンスを使用して Amazon S3 サービスを呼び出します。

Maven は *TODO* コメント付きの次の Handler クラスを生成します。チュートリアルの次のステップでは、*TODO* をコードに置き換えます。

## Maven によって生成された Handler クラス

```
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.
    }
}
```

ロジックを設定するには、Handler クラスのすべてのコンテンツを次のコードに置き換えます。sendRequest メソッドが入力され、必要なインポートが追加されます。

## Handler クラス、実装済み

このコードはまず、バケット名を一意にするために、`System.currentTimeMillis()` を使用して生成された名前の最後の部分を使用して新しい S3 バケットを作成します。

`createBucket()` メソッドでバケットを作成すると、プログラムは `S3Client` の [putObject](#) メソッドを使用してオブジェクトをアップロードします。オブジェクトの内容は、`RequestBody.fromString` メソッドで作成された単純な文字列です。

最後に、プログラムは、`cleanUp` メソッド内のオブジェクトを削除し、その後にバケットを削除します。

```
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");
}
}
```

## ステップ 4: アプリケーションを構築して実行する

プロジェクトが作成され、完全な Handler クラスを含めたら、アプリケーションを構築して実行します。

1. IAM Identity Center セッションがアクティブであることを確認してください。そのためには、AWS Command Line Interface コマンド `aws sts get-caller-identity` を実行してレスポンス

スを確認します。アクティブなセッションがない場合は、[このセクション](#)の手順を参照してください。

2. ターミナルまたはコマンドプロンプトウィンドウを開いて、プロジェクトディレクトリ `getstarted` に移動します。
3. プロジェクトを構築するには、以下のコマンドを使用します。

```
mvn clean package
```

4. 次のコマンドを使用して、アプリケーションを実行します。

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

プログラムが作成した新しいバケットとオブジェクトを表示するには、次のステップを実行します。

1. `Handler.java` で、`sendRequest` メソッド内の `cleanUp(s3Client, bucket, key)` 行をコメントアウトし、ファイルを保存します。
2. `mvn clean package` を実行してプロジェクトを再構築します。
3. `mvn exec:java -Dexec.mainClass="org.example.App"` を再実行してテキストオブジェクトをもう一度アップロードします。
4. [S3 コンソール](#) にサインインして、新しく作成されたバケット内の新しい対象を表示します。

ファイルを表示した後、オブジェクトを削除し、その後にバケットを削除します。

## 成功

Maven プロジェクトがエラーなしで構築および実行された場合は、正常に完了しています! Java 2.x 用 SDK を使用して最初の Java アプリケーションを正常に構築しました。

## クリーンアップ

このチュートリアルで作成されたリソースをクリーンアップするには、以下を行います。

- まだ行っていない場合は、[S3 コンソール](#) で、アプリケーションの実行時に作成されたオブジェクトとバケットをすべて削除します。
- プロジェクトフォルダ (`getstarted`) を削除します。



## 次のステップ

これで基本を理解したので、次の事項を学習する準備が整いました。

- [の使用 Amazon S3](#)
- [、 DynamoDB、 およびさまざまなデータベースサービスなど Amazon EC2、 他の の操作 Amazon Web Services](#)
- [SDK を使用する](#)
- [のセキュリティ AWS SDK for Java](#)

# AWS SDK for Java 2.x のセットアップ

このセクションでは、AWS SDK for Java 2.xを使用する開発環境とプロジェクトについて説明します。

## 設定の概要

AWS のサービス を使用して にアクセスするアプリケーションを正常に開発するには AWS SDK for Java、次の条件が必要です。

- Java SDK は、ユーザーに代わって [リクエストを認証](#) するための認証情報にアクセスできる必要があります。
- SDK 用に設定された [IAM ロールのアクセス許可](#) は、アプリケーション AWS のサービス に必要なへのアクセスを許可する必要があります。PowerUserAccess AWS 管理ポリシーに関連付けられたアクセス許可は、ほとんどの開発ニーズに十分です。
- 以下の要素を備えた開発環境:
  - [共有設定ファイル](#) は、次のいずれかの方法で設定できます。
    - config ファイルには、SDK が AWS 認証情報を取得できるように [IAM Identity Center シングルサインオン設定](#) が含まれています。
    - credentials ファイルには一時的な認証情報が含まれています。
  - [Java 8 以降のインストール](#)。
  - [Maven](#) や [Gradle](#) などの [構築オートメーションツール](#)。
  - コードを使用するテキストエディター。
  - ( オプションですが、推奨 ) [IntelliJ IDEA](https://www.eclipse.org/ide/)、[Eclipse](https://www.eclipse.org/ide/)、または などの IDE (統合開発環境 ) [NetBeans](#)。

IDE を使用すると、 を統合し AWS Toolkitで をより簡単に操作することもできます AWS のサービス。 [AWS Toolkit for IntelliJ](#) と、 [AWS Toolkit for Eclipse](#) は Java 開発に使用できる 2 つのツールキットです。

- アプリケーションを実行する準備ができたときのアクティブな AWS アクセスポータルセッション。を使用して AWS Command Line Interface、IAM Identity Center の AWS アクセスポータルへの [サインインプロセスを開始します](#)。

### ⚠ Important

このセットアップセクションの手順は、ユーザーまたは組織が IAM アイデンティティセンターを使用していることを前提としています。組織が IAM アイデンティティセンターとは独立して機能する外部 ID プロバイダーを使用している場合は、SDK for Java で使用するための一時的な認証情報を取得する方法をご確認ください。[以下の手順](#)に従って、`~/.aws/credentials` ファイルに一時的な認証情報を追加します。

ID プロバイダーが一時的な認証情報を `~/.aws/credentials` ファイルに自動的に追加する場合は、SDK または AWS CLI にプロファイル名を指定する必要がないように、プロファイル名が `[default]` であることを確認してください。

## 認証を設定する。

AWS SDKs [およびツールリファレンスガイドの「認証とアクセス」](#) トピックでは、認証するさまざまなオプションについて説明します。SDK が認証情報を取得できるように、[IAM Identity Center へのアクセスを設定する](#) 手順に従うことをお勧めします。指示に従った後、SDK [がリクエストを認証できるようにシステムが設定されます](#)。

## SDK のシングルサインオンアクセスのセットアップ

SDK が IAM Identity Center 認証を使用できるように、[プログラムによるアクセスセクション](#) のステップ 2 を完了した後、システムには次の要素が含まれている必要があります。

- アプリケーションを実行する前に [AWS アクセスポータルセッション](#) を開始 AWS CLI するために使用する。
- [デフォルトプロファイル](#) を含む `~/.aws/config` ファイル。SDK for Java は、リクエストを AWS に送信する前に、プロファイルの SSO トークンプロバイダー設定を使用して認証情報を取得します。IAM Identity Center アクセス許可セットに接続された IAM ロールである `sso_role_name` 値は、アプリケーションで AWS のサービス 使用されている へのアクセスを許可する必要があります。

次のサンプル config ファイルは、SSO トークンプロバイダー設定で設定されたデフォルトプロファイルを示しています。プロファイルの `sso_session` 設定は、指定された `sso-session` セクションを参照します。`sso-session` セクションには、AWS アクセスポータルセッションを開始するための設定が含まれています。

```
[default]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole
region = us-east-1
output = json

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

SSO トークンプロバイダー設定で使用される設定の詳細については、「AWS SDKs and Tools リファレンスガイド」の「[SSO token provider configuration](#)」を参照してください。

開発環境が前述のようにプログラムによるアクセス用に設定されていない場合は、[SDK リファレンスガイドのステップ 2](#)に従ってください。

## を使用してサインインする AWS CLI

にアクセスするアプリケーションを実行する前に AWS のサービス、SDK が IAM Identity Center 認証を使用して認証情報を解決するために、アクティブな AWS アクセスポータルセッションが必要です。で次のコマンドを実行して AWS CLI、AWS アクセスポータルにサインインします。

```
aws sso login
```

デフォルトのプロファイルを設定している場合は、`--profile` オプションを指定してコマンドを呼び出す必要はありません。SSO トークンプロバイダー設定で名前付きプロファイルを使用している場合、コマンドは `aws sso login --profile named-profile` です。

既にアクティブなセッションがあるかどうかをテストするには、次の AWS CLI コマンドを実行します。

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

このコマンドへの応答により、共有 config ファイルに設定されている IAM Identity Center アカウントとアクセス許可のセットが報告されます。

### Note

既にアクティブな AWS アクセスポータルセッションがあり、 を実行している場合は `aws sso login`、 認証情報を提供する必要はありません。

ただし、 `botocore` に情報へのアクセス許可を求めるダイアログが表示されます。 `botocore` は AWS CLI の基盤です。

許可 を選択して、 AWS CLI および SDK for Java の情報へのアクセスを許可します。

## Java と構築ツールをインストールする

使用する開発環境には次が必要です。

- Java 8 以降。は、 [Oracle Java SE Development Kit](#) と、 Red Hat OpenJDK、 および Adoptium などの Open Java Development Kit (OpenJDK ) のディストリビューションで AWS SDK for Java 動作します。 [Amazon Corretto OpenJDK https://adoptium.net/](https://adoptium.net/)
- Apache Maven、 Gradle、 IntelliJ などの Maven Central をサポートする構築ツールまたは IDE。
  - Maven をインストールして使用方法については、 <https://maven.apache.org/> を参照してください。
  - Gradle をインストールして使用方法については、 <https://gradle.org/> を参照してください。
  - IntelliJ IDEA をインストールして使用方法については、 <https://www.jetbrains.com/idea/> を参照してください。

## 追加認証オプション

プロファイルや環境変数の使用など、 SDK の認証に関するその他のオプションについては、「 SDK とツールのリファレンスガイド」の [「設定 AWS SDKs」](#) の章を参照してください。

## Apache Maven プロジェクトを設定する

[Apache Maven](#) を使用して、 AWS SDK for Java プロジェクトの設定および構築、 [SDK 自体の構築](#) を行うことができます。

### 前提条件

Maven で AWS SDK for Java を使用するには、次のものがが必要です。

- Java 8.0 以降。最新の Java SE Development Kit ソフトウェアは <http://www.oracle.com/technetwork/java/javase/downloads/> からダウンロードできます。AWS SDK for Java は [OpenJDK](#) と Amazon Corretto (Open Java Development Kit (OpenJDK) のディストリビューション) でも機能します。最新の OpenJDK バージョンは <https://openjdk.java.net/install/index.html> からダウンロードしてください。[Corretto ページ](#) から最新の Amazon Corretto 8 または Amazon Corretto 11 バージョンをダウンロードします。
- Apache Maven。Maven をインストールする必要がある場合は、<http://maven.apache.org/> にアクセスしてダウンロードおよびインストールします。

## Maven プロジェクトを作成する

コマンドラインから Maven プロジェクトを作成するには、ターミナルまたはコマンドプロンプトウィンドウ で、次のコマンドを実行します。

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

### Note

com.example.myapp をアプリケーションの完全パッケージ名前空間に置き換えます。また、myapp をプロジェクト名に置き換えます。これがプロジェクトのディレクトリの名前になります。

アーキタイプの最新バージョンを使用するには、**2.X.X** を [Maven 中央の最新バージョン](#) に置き換えてください。

このコマンドにより、アーキタイプテンプレートツールキットを使用して Maven プロジェクトが作成されます。アーキタイプは AWS Lambda 関数ハンドラープロジェクトの雛形を生成します。このプロジェクトのアーキタイプは、Java SE 8 でコンパイルするように事前設定されており、-DarchetypeVersion で指定された Java 2.x 用 SDK のバージョンへの依存関係が含まれています。

Maven プロジェクトの作成と設定の詳細については、「[Maven 入門ガイド](#)」を参照してください。

## Maven 用の Java コンパイラを設定する

前述のように AWS Lambda プロジェクトアーキタイプを使用してプロジェクトを作成した場合、Java コンパイラの設定はすでに完了しています。

この設定が存在することを確認するには、まず、前のコマンドを実行したときに作成したプロジェクトフォルダ (myapp など) から pom.xml ファイルを開きます。11 行目と 12 行目を見て、この Maven プロジェクトの Java コンパイラのバージョン設定を確認します。また、71~75 行目を見て、必要な Maven コンパイラプラグインが含まれていることを確認します。

```
<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>
```

別のアーキタイプで、または別の方法を使用してプロジェクトを作成する場合は、Maven コンパイラプラグインがビルドの一部となっており、そのソースプロパティとターゲットプロパティの両方が pom.xml ファイルで 1.8 に設定されているようにする必要があります。

これらの必須の設定を指定する 1 つの方法として、前述のスニペットを参照してください。

または、次のように、プラグイン宣言を使用してインラインでコンパイラ設定を指定することもできます。

```
<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>
```

## SDK を依存関係として宣言する

プロジェクトで AWS SDK for Java を使用するには、プロジェクトの pom.xml ファイルで SDK を依存関係として宣言する必要があります。

前述のようにプロジェクトアーキタイプを使用してプロジェクトを作成した場合、SDK の最新バージョンはすでにプロジェクトの依存関係として設定されています。

アーキタイプによって software.amazon.awssdk グループ ID の BOM (部品表) アーティファクト依存関係が生成されます。BOM を使用すると、同じグループ ID を共有する個々のアーティファクトの依存関係に Maven バージョンを指定する必要がありません。

別の方法で Maven プロジェクトを作成した場合は、pom.xml ファイルに次の内容が含まれていることを確認して、プロジェクトの最新バージョンの SDK を設定します。

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



**Note**

pom.xml ファイル内の **2.X.X** を [最新バージョンの AWS SDK for Java 2.x](#) に置き換えます。

## SDK モジュールの依存関係を設定する

これで SDK を設定したので、プロジェクトで使用する 1 つ以上の AWS SDK for Java モジュールの依存関係を追加できます。

各コンポーネントのバージョン番号を指定できますが、部品表アーティファクトを使用する dependencyManagement セクションで SDK バージョンをすでに宣言しているため、その必要はありません。特定のモジュールの異なるバージョンをロードするには、その依存関係のバージョン番号を指定します。

前述のように、プロジェクトのアーキタイプを使用してプロジェクトを作成した場合、プロジェクトはすでに複数の依存関係で設定されています。これらには、次のような AWS Lambda 関数ハンドラーと Amazon S3 の依存関係が含まれます。

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

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

### Note

上の pom.xml 例では、依存関係は異なる groupId からのものです。s3 依存関係は software.amazon.awssdk からのもので、aws-lambda-java-core 依存関係は com.amazonaws からのものです。BOM 依存関係管理設定は software.amazon.awssdk のアーティファクトに影響するため、aws-lambda-java-core アーティファクトにはバージョンが必要です。

Java 2.x 用の SDK を使用して Lambda 関数ハンドラーを開発する場合は、aws-lambda-java-core が正しい依存関係です。ただし listFunctions、deleteFunction、invokeFunction および createFunction などのオペレーションを使用して Lambda リソースを管理する必要があるアプリケーションには、次の依存関係が必要です。

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

### Note

s3 依存関係には、netty-nio-client および apache-client 推移的な依存関係は含まれません。アーキタイプには、これらの HTTP クライアントの代わりに url-connection-client 依存関係が含まれるため、[AWS Lambda 関数の起動レイテンシーを短縮](#)できます。

プロジェクトに必要な AWS のサービス および機能用のモジュールをプロジェクトに追加します。AWS SDK for Java BOM によって管理されるモジュール (依存関係) は [Maven の中央リポジトリ](#)に一覧表示されます。

**Note**

コード例から pom.xml ファイルを参照して、プロジェクトに必要な依存関係を特定できます。たとえば、DynamoDB サービスの依存関係に関心がある場合は、GitHub の「[AWS Code Examples Repository](#)」にある「[This example](#)」を参照してください。( [/javav2/example\\_code/dynamodb](#) にある pom.xml ファイルを確認してください。)

## SDK 全体のプロジェクトへのビルド

アプリケーションを最適化するため、SDK 全体ではなく必要なコンポーネントのみインポートすることを強くお勧めします。ただし、AWS SDK for Java 全体をプロジェクトにビルドするには、次のように pom.xml ファイルで宣言します。

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

## プロジェクトを構築する

pom.xml ファイルを設定したら、Maven を使用してプロジェクトをビルドできます。

コマンドラインから Maven プロジェクトを構築するには、ターミナルウィンドウまたはコマンドプロンプトウィンドウを開き、プロジェクトディレクトリ (myapp など) に移動し、次のコマンドを入力するか、貼り付けて Enter キーまたは Return キーを押します。

```
mvn package
```

これにより、target ディレクトリに 1 つの .jar ファイル (JAR) が作成されます (例: myapp/target)。この JAR には、pom.xml ファイルの依存関係として指定したすべての SDK モジュールが含まれています。

# Gradle プロジェクトを設定する

[Gradle](#) を使用して、AWS SDK for Java プロジェクトを設定および構築できます。

次の例での最初のステップは、「[Gradle's Getting Started guide](#)」バージョン 8.4 用からのものです。別のバージョンを使用すると、結果が多少異なる場合があります。

Gradle で Java アプリケーションを作成する方法 (コマンドライン)

1. プロジェクトを格納するディレクトリを作成します。この例では、demo がディレクトリ名です。
2. demo ディレクトリ内で `gradle init` コマンドを実行し、次のコマンドライン出力に示すように、赤で強調表示されている値を指定します。ここでは、ビルドスクリプト DSL 言語として「Kotlin」を選択しますが、このトピックの最後には「Groovy」の完全な例も示されています。

```
> 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` タスクが完了すると、`demo` ディレクトリには次のツリー構造を含みます。次のセクションでは、メインのビルドファイル、`build.gradle.kts` (赤でハイライト) を詳しく見ていきます。

```
### 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 ファイルには次の雛形となるコンテンツが含まれます。

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

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

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

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

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

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

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


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

```
}

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

4. 雛形となる Gradle ビルドファイルを AWS プロジェクトの基礎として使用してください。
  - a. Gradle プロジェクトの SDK 依存関係を管理するには、AWS SDK for Java 2.x の Maven 部品表 (BOM) を build.gradle.kts ファイルの dependencies セクションに追加します。

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

 Note

このビルドファイル例では、2.21.1 を Java 2.x 用の SDK の最新バージョンに置き換えます。[Maven 中央リポジトリ](#)で入手可能な最新バージョンを確認してください。

- b. アプリケーションが必要とする SDK モジュールを dependencies セクションで指定してください。例として、以下では Amazon Simple Storage Service への依存関係を追加します。

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

Gradle は BOM の情報を使用して、正しいバージョンの宣言された依存関係を自動的に解決します。

以下の例は、Kotlin と Groovy DSL の両方における完全な Gradle ビルドファイルを示しています。ビルドファイルには、Amazon S3、認証、ログ記録、テストの依存関係が含まれています。Java のソースおよびターゲットバージョンはバージョン 11 です。

#### Kotlin DSL (build.gradle.kts)

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

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

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

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



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

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

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

## Groovy DSL (build.gradle)

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

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

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

dependencies {
    implementation platform('software.amazon.awssdk:bom:2.21.1')
```

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

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

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

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

次のステップについては、Gradle ウェブサイトの「入門ガイド」で、[Gradle アプリケーションをビルドして実行する](#)方法を確認してください。

## AWS SDK for Java の GraalVM Native Image プロジェクトを設定する

バージョン 2.16.1 以降では、AWS SDK for Java は GraalVM Native Image アプリケーションにすぐに使用できるサポートを提供します。archetype-app-quickstart Maven アーキタイプを使用して、ネイティブイメージサポートが組み込まれたプロジェクトを設定します。

### 前提条件

- [AWS SDK for Java 2.x の設定](#)の手順を完了します。

- [GraalVM Native Image](#) をインストールします。

## アーキタイプを使用してプロジェクトを作成する

ネイティブイメージサポートが組み込まれた Maven プロジェクトを作成するには、ターミナルまたはコマンドプロンプトウィンドウで、次のコマンドを使用します。

### Note

アプリケーションのフルパッケージ名前空間を `com.example.mynativeimageapp` に置き換えます。また、`mynativeimageapp` をプロジェクト名に置き換えます。これがプロジェクトのディレクトリの名前になります。

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

このコマンドは、AWS SDK for Java、Amazon S3、および ApacheHttpClient HTTP クライアントの依存関係で設定された Maven プロジェクトを作成します。また、[GraalVM Native Image Maven プラグイン](#)への依存関係も含まれているため、Maven を使用してネイティブイメージを構築できます。

別の Amazon Web Services の依存関係を含めるには、`-Dservice` パラメータの値をそのサービスのアーティファクト ID に設定します。例には、`dynamodb`、`comprehend`、`pinpoint` が含まれます。アーティファクト ID の詳細なリストについては、[Maven Central の software.amazon.awssdk](#) のマネージド依存関係のリストを参照してください。

非同期 HTTP クライアントを使用するには、`-DhttpClient` パラメータを `netty-nio-client` に設定します。同期 HTTP クライアントとして `apache-client` の代わりに `URLConnectionHttpClient` を使用するには、`-DhttpClient` パラメータを `url-connection-client` に設定します。

## ネイティブイメージを構築する

プロジェクトを作成したら、プロジェクトディレクトリから次のコマンドを実行します (例: mynativeimageapp)。

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

これにより、target ディレクトリにネイティブイメージアプリケーションが作成されます (例: target/mynativeimageapp)。

# を使用する AWS SDK for Java 2.x

SDK のセットアップのステップを完了すると、Amazon S3、DynamoDB、Amazon EC2 などの AWS サービスにリクエストを行う準備が整います。

## サービスクライアントを使用する

### サービスクライアントを作成する

にリクエストを行うには AWS のサービス、まず静的ファクトリメソッドを使用して、そのサービスのサービスクライアントをインスタンス化する必要があります `builder()`。 `builder()` メソッドは、サービスクライアントをカスタマイズできる `builder` オブジェクトを返します。 fluent setter メソッドでは `builder` オブジェクトが返るため、利便性とコードの可読性を高めるためにメソッドの呼び出しをチェーンすることができます。必要なプロパティを設定した後、 `build()` メソッドを呼び出してクライアントを作成します。

例として、以下のコードスニペットは、 `Ec2Client` オブジェクトを Amazon EC2 のサービスクライアントとしてインスタンス化します。

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

#### Note

SDK のサービスクライアントはスレッドセーフです。最高のパフォーマンスを得るには、長期間使用するオブジェクトとしてこれら进行处理します。各クライアントには、クライアントがガーベージコレクションの対象になった場合にリリースされる独自の接続プールリソースを備えています。

サービスクライアントオブジェクトはイミュータブルであるため、リクエストを実行する各サービスについて、または同じサービスにリクエストを実行するために異なる設定を使用する場合は、新しいクライアントを作成する必要があります。

サービスクライアントビルダー `Region` で を指定することは、すべての AWS サービスに必須ではありませんが、アプリケーションで実行する API コールのリージョンを設定するのがベストプラクティスです。詳細については、「[AWS リージョンの選択](#)」を参照してください。

## デフォルトのクライアント設定

このクライアントビルダーには、`create()` という名前の別のファクトリメソッドがあります。このメソッドでは、サービスクライアントがデフォルト設定で作成されます。デフォルトのプロバイダーチェーンを使用して認証情報と AWS リージョンをロードします。アプリケーションを実行している環境から認証情報またはリージョンを決定できない場合、`create` の呼び出しは失敗します。SDK が認証情報および使用するリージョンを決定する方法の詳細については、「[認証情報の使用](#)」および「[リージョンの選択](#)」を参照してください。

例えば、以下のコードスニペットは、`DynamoDbClient` オブジェクトを Amazon DynamoDB のサービスクライアントとしてインスタンス化します。

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

## サービスクライアントを設定する

サービスクライアントの設定をカスタマイズするには、`builder()` ファクトリメソッドのセッターを使用します。便宜上、より読みやすいコードを作成するために、メソッドをつなぎ合わせて複数の構成オプションを設定します。

以下の例は、複数のカスタム設定で構成された `S3Client` を示しています。

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

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

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

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

## リクエストを発行する

サービスクライアントを使用して、対応する `runInstances` にリクエストを行います AWS のサービス。

例えば、このコードスニペットは、`RunInstancesRequest` オブジェクトを作成して新しい Amazon EC2 インスタンスを作成する方法を示しています。

```
// 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 は、リクエストを作成してインスタンスに渡すのではなく、リクエストの作成に使用できるビルダーを備えています。ビルダーを使用すると、Java Lambda 式を使用してリクエストを「インライン」で作成できます。

次の例では、[ビルダーを使用して](#)リクエストを作成する `runInstances` メソッドのバージョンを使用して、前の例を書き換えています。

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

## レスポンスを処理する

SDK は、ほとんどのサービスオペレーションのレスポンスオブジェクトを返します。コードは、必要に応じてレスポンスオブジェクト内の情報を処理できます。

例えば、次のコードスニペットは、前のリクエストの [RunInstancesResponse](#) オブジェクトで返された最初のインスタンス ID を出力します。

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

ただし、すべてのオペレーションがサービス固有のデータを含むレスポンスオブジェクトを返すわけではありません。このような状況では、HTTP レスポンスのステータスをクエリして、オペレーションが成功したかどうかを確認できます。

例えば、次のスニペットのコードは、HTTP レスポンスをチェックして、Amazon Simple Email Service の [DeleteContactList](#) オペレーションが成功したかどうかを確認します。

```
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());
}
```

## サービスクライアントを閉じる

ベストプラクティスとして、アプリケーションの存続期間中の複数の API サービス呼び出しにはサービスクライアントを使用する必要があります。ただし、サービスクライアントを 1 回だけ使用する必要がある場合や、サービスクライアントが不要になったら閉じます。

サービスクライアントが不要になったら、`close()` メソッドを呼び出してリソースを解放します。

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

サービスクライアントを 1 回だけ使用する必要がある場合は、`try-with-resources` ステートメントでサービスクライアントをリソースとしてインスタンス化できます。サービスクライアントは [Autoclosable](#) インターフェースを実装するため、JDK はステートメントの最後で自動的に `close()` メソッドを呼び出します。



以下の例は、サービスクライアントを使用して 1 回限りのコールを実行する方法を示しています。を呼び出す `StsClient` は、AWS Security Token Service は、アカウント ID を返した後で閉じられます。

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

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

## 例外を処理する

SDK はランタイムの (またはチェックされていない) 例外を使用して、エラー処理をきめ細かく制御し、例外処理がアプリケーションに合わせてスケールされるようにします。

[SdkServiceException](#)、またはそのサブクラスの 1 つは、SDK がスローする最も一般的な例外フォームです。これらの例外は、AWS のサービスからの応答を表します。また、ネットワーク接続の障害など、クライアント側 (つまり、開発環境やアプリケーション環境) で問題が発生した場合に生じる [SdkClientException](#) を処理することもできます。

このコードスニペットは、ファイルを Amazon S3 にアップロードするときサービスの例外を処理する 1 つの方法を示しています。このコード例は、クライアントとサーバーの両方の例外を受け取り、詳細をログに記録して、アプリケーションを存在させます。

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

try {

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

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

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

詳細については、[例外の処理](#)を参照してください。

## ウェイターを使用する

で新しいテーブルを作成したり、新しい Amazon S3 バケットを作成 DynamoDB したりするなど、処理に時間がかかるリクエストもあります。コードの実行を続行する前にリソースの準備ができているようにするには、Waiter を使用します。

例えば、このコードスニペットは に新しいテーブル (myTable) を作成し DynamoDB、テーブルが ACTIVE ステータスになるまで待機してから、レスポンスを出力します。

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

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

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

詳細については、[ウェイターの使用](#)を参照してください。

## HTTP クライアント

AWS SDK for Java を使用して構築するアプリケーションで、HTTP クライアントのデフォルト設定を変更できます。HTTP クライアントと設定を構成する方法については、[HTTP 設定](#)を参照してください。

## 再試行

再試行モードやバックオフ戦略など、サービスクライアントでの再試行のデフォルト設定を変更できません。詳細については、AWS SDK for Java API リファレンスの [RetryPolicy クラス](#) を参照してください。

AWS サービスの再試行の詳細については、[「でのエラーの再試行とエクスポネンシャルバックオフ AWS」](#) を参照してください。

## タイムアウト

[apiCallTimeout](#) および の [apiCallAttemptTimeout](#) セッターを使用して、各サービスクライアントのタイムアウトを設定できます。 [ClientOverrideConfiguration.Builder](#)。 `apiCallTimeout` の設定は、クライアントが API コールの実行を完了するまでの猶予時間です。 `apiCallAttemptTimeout` 設定は、各 HTTP リクエスト (再試行) が完了してからあきらめるまでの待機時間です。

次の例では、S3 クライアントの両方のタイムアウトを設定します。

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

また、 `apiCallTimeout` を設定し、 `overrideConfiguration` メソッドを使用してリクエストオブジェクトに提供することで [AwsRequestOverrideConfiguration](#)、リクエストレベルでタイムアウトを設定することもできます。

次の例では、同じタイムアウト設定を使用しますが、S3 PutObject オペレーションのリクエストレベルで使用します。

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

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

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

## 実行インターセプター

リクエスト/応答のライフサイクルのさまざまな部分で、API リクエストと応答の実行をインターセプトするコードを記述できます。これにより、メトリクスの発行、実行中のリクエストの変更、リクエスト処理のデバッグ、例外の表示などを行うことができます。詳細については、AWS SDK for Java「API リファレンス」の「[ExecutionInterceptor インターフェイス](#)」を参照してください。

## 追加情報

- 上記のコードスニペットの完全な例については、[「 の使用 Amazon DynamoDB」](#)、[「 の使用 Amazon EC2」](#)、[「 の使用 Amazon S3」](#)を参照してください。

## SDK への一時的な認証情報の提供

Amazon Web Services を使用して にリクエストを行う前に AWS SDK for Java 2.x、SDK は によって発行された一時的な認証情報を暗号で署名します AWS。一時的な認証情報にアクセスするために、SDK は複数の場所を確認して設定値を取得します。

このトピックでは、SDK が一時的な認証情報にアクセスできるようになるいくつかの方法について説明します。

### トピック

- [一時的な認証情報へのアクセスを設定する](#)
- [デフォルトの認証情報プロバイダーチェーン](#)
- [特定の認証情報プロバイダーまたはプロバイダーチェーンを使用する](#)
- [プロファイルを使用する](#)
- [外部プロセスからの一時的な認証情報をロードする](#)
- [コードに一時的な認証情報を入力する](#)

## • [での IAM ロール認証情報の読み取り Amazon EC2](#)

### 一時的な認証情報へのアクセスを設定する

セキュリティを強化するために、SDK for Java では存続期間の長い[認証情報の代わりに一時的な認証情報を使用する](#)ように設定 AWS することをお勧めします。一時的な認証情報は、アクセスキー (アクセスキー ID とシークレットアクセスキー) およびセッショントークンで構成されています。トークンの更新プロセスは自動的に行われるため、一時的な認証情報を自動的に取得するように [SDK を設定](#) することをお勧めします。ただし、[SDK に一時的な認証情報を直接提供](#) することはできません。

### IAM Identity Center の設定

このガイドの [???](#) で説明されているように IAM Identity Center シングルサインオンアクセスを使用するように SDK を設定すると、SDK は自動的に一時的な認証情報を使用します。

SDK は IAM Identity Center アクセストークンを使用して、config ファイルの `sso_role_name` 設定で設定された IAM ロールにアクセスします。SDK はこの IAM ロールを委任され、AWS のサービスリクエストに使用する一時的な認証情報を取得します。

SDK が設定から一時的な認証情報を取得する方法の詳細については、「[SDK とツールリファレンスガイド](#)」の「[IAM Identity Center 認証について AWS SDKs](#)」セクションを参照してください。

### AWS アクセスポータルから取得する

IAM Identity Center シングルサインオン設定の代わりに、AWS アクセスポータルで利用できる一時的な認証情報をコピーして使用できます。一時的な認証情報は、プロファイルで使用することも、システムプロパティや環境変数の値として使用することもできます。

一時的な認証情報用のローカルの認証情報ファイルを設定する

1. [認証情報の共有ファイルを作成](#) する
2. 認証情報ファイルに、作業用の一時的な認証情報を貼り付けるまで、次のプレースホルダーテキストを貼り付けます。

```
[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>
```



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

## 認証情報の設定の取得順序

SDK for Java 2.x のデフォルトの認証情報プロバイダーチェーンは、事前定義されたシーケンスを使用して、環境内の設定を検索します。

### 1. Java のシステムプロパティ

- SDK は [SystemPropertyCredentialsProvider](#) クラスを使用して、`aws.accessKeyId`、`aws.secretAccessKey`および `aws.sessionTokenJava` システムプロパティから一時的な認証情報をロードします。

#### Note

Java システムプロパティの設定方法の詳細については、Java Tutorials の公式ウェブサイトにある[システムプロパティ](#)のチュートリアルを参照してください。

### 2. 環境変数

- SDK は [EnvironmentVariableCredentialsProvider](#) クラスを使用して、`AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY`および `AWS_SESSION_TOKEN`環境変数から一時的な認証情報をロードします。

### 3. からのウェブ ID トークン AWS Security Token Service

- SDK は [WebIdentityTokenFileCredentialsProvider](#) クラスを使用して、Java システムプロパティまたは環境変数から一時的な認証情報をロードします。

### 4. 共有 credentials および config ファイル

- SDK は [ProfileCredentialsProvider](#)を使用して、共有 credentials および config ファイルの[default]プロファイルから IAM Identity Center シングルサインオン設定または一時的な認証情報をロードします。

AWS SDKs およびツールリファレンスガイドには、SDK for Java が IAM Identity Center シングルサインオントークンと連携して、SDK が を呼び出すために使用する一時的な認証情報を取得する方法に関する詳細[情報](#)が記載されています AWS のサービス。

**Note**

credentials および config ファイルは、さまざまな AWS SDKs および ツールによって共有されます。詳細については、「AWS SDKs and Tools リファレンスガイド」の「[.aws/credentials and .aws/config files](#)」を参照してください。

## 5. Amazon ECS コンテナ認証情報

- SDK は [ContainerCredentialsProvider](#) クラスを使用して、AWS\_CONTAINER\_CREDENTIALS\_RELATIVE\_URI システム環境変数から一時的な認証情報をロードします。

## 6. Amazon EC2 IAM ロールから提供された インスタンスの認証情報

- SDK は [InstanceProfileCredentialsProvider](#) クラスを使用して、Amazon EC2 メタデータサービスから一時的な認証情報をロードします。

## 特定の認証情報プロバイダーまたはプロバイダーチェーンを使用する

デフォルトの認証情報プロバイダーチェーンの代わりに、SDK が使用する認証情報プロバイダーを指定することもできます。特定の認証情報プロバイダーを指定すると、SDK はさまざまな場所を確認するプロセスをスキップするため、サービスクライアントの作成にかかる時間がわずかに短縮されます。

例えば、環境変数を使用してデフォルト設定を設定する場合は、次のコードスニペットのように、サービスクライアントビルダーの `credentialsProvider` メソッドに [EnvironmentVariableCredentialsProvider](#) オブジェクトを指定します。

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

認証情報プロバイダーとプロバイダーチェーンの完全なリストについては、「[のすべての既知の実装クラス](#)」を参照してください [AwsCredentialsProvider](#)。



**Note**

AwsCredentialsProvider インターフェイスを実装することにより、独自の認証情報プロバイダーまたはプロバイダーチェーンを使用できます。

## プロファイルを使用する

共有 config と credentials ファイルを使用して、複数のプロファイルを設定できます。これにより、アプリケーションは複数の認証情報設定を使用できます。[default] プロファイルについては前述に記載したとおりです。SDK は [ProfileCredentialsProvider](#) クラスを使用して、共有 credentials ファイルで定義されたプロファイルから設定をロードします。

次のコードスニペットは、my\_profile というプロファイルの一部として定義された認証情報を使用するサービスクライアントを構築する方法を示しています。

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

### 別のプロファイルをデフォルトとして設定する

[default] プロファイル以外のプロファイルをアプリケーションのデフォルトとして設定するには、AWS\_PROFILE 環境変数をカスタムプロファイルの名前に設定します。

この変数を Linux、macOS、または Unix で設定するには、export を使用します。

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

Windows でこれらの変数を設定するには、set を使用します。

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

または、aws.profile Java システムプロパティをプロファイルの名前に設定します。

## プロファイルの認証情報を再ロード

ビルダーにプロファイル認証情報を再ロードする `profileFile()` メソッドを持つ認証情報プロバイダーならどれでも設定できます。これらの認証情報プロファイルクラスは `ProfileCredentialsProvider`、`DefaultCredentialsProvider`、`InstanceProfileCredential` および `ProfileTokenProvider` です。

### Note

プロファイル認証情報の再ロードは、プロファイルファイル内の設定でのみ機能します。aws\_access\_key\_id、aws\_secret\_access\_key、および aws\_session\_token、region、sso\_session、sso\_account\_id、および source\_profile などの設定は無視されます。

サポートされている認証情報プロバイダーがプロファイル設定を再ロードするように設定するには、`profileFile()` ビルダーメソッドに [ProfileFileSupplier](#) のインスタンスを指定します。次のコード例は、[default] プロファイルから認証情報設定を再ロードする `ProfileCredentialsProvider` を示しています。

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

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

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

`ProfileCredentialsProvider.resolveCredentials()` が呼び出されると、SDK for Java は設定を再ロードします。`ProfileFileSupplier.defaultSupplier()` は SDK が提供する `ProfileFileSupplier` の [いくつかの便利な実装](#) の 1 つです。ユースケースで必要な場合は、独自の实装を提供できます。

次の例は、`ProfileFileSupplier.reloadWhenModified()` の便利なメソッドの使用法を示しています。`reloadWhenModified()` は `Path` パラメータを取るため、標準 `~/.aws/credentials` (または `config`) 場所ではなく、設定のソースファイルを柔軟に指定できます。

ファイルの内容が変更されると SDK が判断した場合にのみ、`resolveCredentials()` が呼び出されたときに設定が再ロードされます。

```
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()` メソッドは複数の設定ファイルの内容をマージします。`resolveCredentials()` への呼び出しごとにファイルを再ロードするか、ファイルの設定を最初に読み取った時点で固定するかを決めることができます。

次の例は、プロファイル設定を含む 2 つのファイルの設定を結合する `DefaultCredentialsProvider` を示しています。SDK は、`resolveCredentials()` が呼び出されて設定が変更されるたびに、`credentialsFilePath` 変数が指すファイル内の設定を再ロードします。`profileFile` オブジェクトの設定は変わりません。

```
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.  
*/
```

## 外部プロセスからの一時的な認証情報をロードする

### Warning

以下では、外部プロセスから一時的な認証情報を調達する方法について説明します。これは危険な場合があるため、注意して進めてください。可能であれば、他の認証情報プロバイダーの利用をお勧めします。このオプションを使用する場合は、使用しているオペレーティングシステムのセキュリティ上のベストプラクティスに従って、config ファイルができるだけロックされていることを確認する必要があります。

カスタム認証情報ツールが StdErr に秘密情報を書き込まないようにしてください。SDKs と AWS CLI は、このような情報をキャプチャしてログに記録し、不正ユーザーに公開する可能性があります。

SDK for Java 2.x では、カスタムユースケース用に外部プロセスから一時的な認証情報を取得できます。この機能を設定するには、次の 2 つの方法があります。

### credential\_process 設定を使用する

一時的な認証情報を提供する方法がある場合は、その credential\_process 設定を config ファイル内のプロファイル定義の一部として追加することで統合できます。指定する値には、コマンドファイルへのフルパスを使用する必要があります。ファイルパスにスペースが含まれている場合は、引用符で囲む必要があります。

SDK は指定されたとおりにコマンドを呼び出し、stdout から JSON データを読み取ります。

次の例は、スペースを含まないファイルパスとスペースを含むファイルパスにこの設定を使用する方法を示しています。

#### Linux/macOS

ファイルパス内にスペースを含まない

```
[profile process-credential-profile]
```

```
credential_process = /path/to/credential/file/credential_file.sh --custom-command
custom_parameter
```

### ファイルパス内のスペースを含む

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

## Windows

### ファイルパス内にスペースを含まない

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

### ファイルパス内のスペースを含む

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

次のコードスニペットは、`process-credential-profile` というプロファイルの一部として定義された一時的な認証情報を使用するサービスクライアントを構築する方法を示しています。

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

外部プロセスを一時的な認証情報のソースとして使用方法の詳細については、AWS SDKs [「プロセス認証情報」セクション](#)を参照してください。

## ProcessCredentialsProvider を使用する

config ファイル内の設定を使用する代わりに、SDK の [ProcessCredentialsProvider](#) を使用して Java を使用して一時的な認証情報を読み込むこともできます。

以下の例は、`ProcessCredentialsProvider` を使用して外部プロセスを指定する方法と、一時的な認証情報を使用するサービスクライアントを設定する方法のさまざまなバージョンを示しています。

## Linux/macOS

### ファイルパス内にスペースを含まない

```
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();
```

### ファイルパス内のスペースを含む

```
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

### ファイルパス内にスペースを含まない

```
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();
```

## ファイルパス内のスペースを含む

```
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();
```

## コードに一時的な認証情報を入力する

デフォルトの認証情報チェーン、または特定あるいはカスタムのプロバイダーやプロバイダーチェーンがアプリケーションに対して機能しない場合は、一時的な認証情報をコードで直接指定できます。これらは、[上記の IAM ロール認証情報](#)、または AWS Security Token Service () から取得した一時的な認証情報です AWS STS。を使用して一時的な認証情報を取得した場合は AWS STS、次のコード例に示すように、それらを AWS のサービス クライアントに提供します。

1. `StsClient.assumeRole()` を呼び出してロールを委任されます。
2. [StaticCredentialsProvider](#) オブジェクトを作成し、`AwsSessionCredentials` オブジェクトを指定します。
3. `StaticCredentialsProvider` を使用してサービスクライアントビルダーを設定し、クライアントを構築します。

次の例では、IAM が引き受けたロール AWS STS の から返された一時的な認証情報を使用して Amazon S3 サービスクライアントを作成します。

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

## 許可セット

AWS IAM Identity Center で定義されている次のアクセス許可セットにより、アイデンティティ (ユーザー) は次の 2 つのオペレーションを実行できます。

1. Amazon Simple Storage Service の `GetObject` オペレーション。
2. AWS Security Token Service の `AssumeRole` オペレーション。

ロールを引き受けない場合は、この例に示した `s3.listBuckets()` メソッドは失敗します。

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

## 役割を引き受けた

### 委任されたロールのアクセス許可ポリシー

前の例で委任されたロールには、次のアクセス許可ポリシーがアタッチされています。このアクセス許可ポリシーでは、ロールと同じアカウントのすべてのバケットを一覧表示できます。

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

```
{
  "Effect": "Allow",
  "Action": [
    "s3:ListAllMyBuckets"
  ],
  "Resource": [
    "*"
  ]
}
```

## 委任されたロールの信頼ポリシー

次の信頼ポリシーは、前の例で委任されたロールにアタッチされています。このポリシーでは、2つのアカウントの ID (ユーザー) がロールを委任されることができます。

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

## での IAM ロール認証情報の読み取り Amazon EC2

IAM ロールを使用して、EC2 インスタンスで実行され、AWS CLI または AWS API リクエストを作成しているアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。EC2 インスタンスに AWS ロールを割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」

の「[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して権限を付与する](#)」を参照してください。

このトピックでは、EC2 インスタンスで実行するように Java アプリケーションをセットアップし、SDK for Java で IAM ロール認証情報を取得できるようにする方法について説明します。

## 環境から IAM ロールの認証情報を取得する

アプリケーションが `create` メソッド (または `builder().build()` メソッド) を使用して AWS サービスクライアントを作成する場合、SDK for Java はデフォルトの認証情報プロバイダーチェーンを使用します。デフォルトの認証情報プロバイダーチェーンは、SDK が一時的な認証情報と取引できる設定要素を実行環境を検索します。[the section called “デフォルトの認証情報プロバイダーチェーン”](#) セクションでは、検索プロセス全体について説明します。

デフォルトのプロバイダーチェーンの最後のステップは、アプリケーションが Amazon EC2 インスタンスで実行されている場合にのみ使用できます。このステップでは、SDK は `InstanceProfileCredentialsProvider` を使用して EC2 インスタンスプロファイルで定義されている IAM ロールを読み取ります。次に SDK はその IAM ロールの一時的な認証情報を取得します。

これらの認証情報は一時的なもので、最終的には失効しますが、`InstanceProfileCredentialsProvider` によって定期的に更新されるため、AWS にアクセスを許可し続けることができます。

## IAM ロールの認証情報をプログラムで取得する

`InstanceProfileCredentialsProvider` EC2 を最終的に使用するデフォルトの認証情報プロバイダーチェーンの代わりに、`InstanceProfileCredentialsProvider` を使用してサービスクライアントを明示的に設定できます。このアプローチは次のスニペットに示されています。

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

## IAM ロールの認証情報を安全に取得する

デフォルトでは、EC2 インスタンスは [IMDS](#) (インスタンスメタデータサービス) を実行します。これにより、SDK は設定された IAM ロールなどの情

報InstanceProfileCredentialsProviderにアクセスできます。EC2 インスタンスはデフォルトで2つのバージョンのIMDSを実行します。

- インスタンスメタデータサービスバージョン 1 (IMDSv1) – リクエスト/レスポンスメソッド
- インスタンスメタデータサービスバージョン 2 (IMDSv2) – セッション指向メソッド

[IMDSv2 は IMDSv1 よりも安全なアプローチ](#)です。IMDSv1

デフォルトでは、Java SDK はまず IMDSv2 を試行して IAM ロールを取得しますが、失敗した場合は IMDSv1 を試行します。ただし、IMDSv1 は安全性が低いため、AWS では、IMDSv2 のみを使用し、SDK による IMDSv1 の試行を無効にすることをお勧めします。

より安全なアプローチを使用するには、次のいずれかの設定に の値を指定して、SDK による IMDSv1 の使用を無効にしますtrue。

- 環境変数: AWS\_EC2\_METADATA\_V1\_DISABLED
- JVM システムプロパティ: aws.disableEc2MetadataV1
- 共有設定ファイルの設定: ec2\_metadata\_v1\_disabled

これらの設定のいずれかを に設定した場合true、最初の IMDSv2 呼び出しが失敗しても、SDK は IMDSv1 を使用して IMDSv2 ロールの認証情報をロードしません。

## の使用 AWS リージョン

AWS リージョン を使用すると、サービスクライアント AWS のサービス は、特定の地理的領域に物理的に存在する にアクセスできます。

### AWS リージョンを明示的に設定する

リージョンを明示的に設定するには、[Region](#) クラスで定義されている定数を使用することをお勧めします。これは、公開されている利用可能なすべてのリージョンを列挙したものです。

そのクラスで列挙されたリージョンが設定されたクライアントを作成するには、クライアントビルダーの region メソッドを使用します。

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

使用するリージョンが Region クラスの列挙でない場合は、静的な of メソッドを使用して新しいリージョンを作成することができます。このメソッドを使用することで、SDK をアップグレードせずに、新しいリージョンにアクセスすることができます。

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

### Note

ビルダーを使用してクライアントを構築した後は、そのクライアントはイミュータブルであり、AWS リージョンを変更することはできません。同じサービス AWS リージョンで複数のを使用する必要がある場合は、リージョンごとに 1 つずつ、複数のクライアントを作成する必要があります。

## SDK による環境に基づくリージョンの自動的な決定

コードが Amazon EC2 または で実行されている場合 AWS Lambda、コードが実行されている AWS リージョン のと同じ を使用するようにクライアントを設定することもできます。これにより、実行中の環境からコードが切り離され、アプリケーションを複数の にデプロイしてレイテンシーや冗長性 AWS リージョン を低く抑えることができます。

認証情報/リージョンのデフォルトプロバイダーチェーンを使用して環境からリージョンを決定するには、クライアントビルダーの create メソッドを使用します。

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

region メソッド AWS リージョン を使用して を明示的に設定しない場合、SDK はデフォルトのリージョンプロバイダーチェーンを参照して、使用するリージョンを決定します。

### デフォルトリージョンプロバイダーチェーンを理解する

SDK は次の手順で AWS リージョン を検索します。

1. ビルダー自体で region を使用して設定されている明示的なリージョンは、その他の設定より優先されます。

2. `AWS_REGION` 環境変数が確認されます。設定されている場合は、そのリージョンを使用してクライアントが設定されます。

#### Note

Lambda コンテナはこの環境変数を設定します。

3. SDK は、AWS 共有設定ファイルと共有認証情報ファイル (通常は `~/.aws/config` と `~/.aws/credentials`) をチェックします。region プロパティが存在する場合、SDK はそれを使用します。
  - SDK が同じプロファイル (プロファイルを含む default) の両方のファイルで region プロパティを見つけた場合、SDK は共有認証情報ファイル内の値を使用します。
  - `AWS_CONFIG_FILE` 環境変数を使用すると、共有設定ファイルの場所をカスタマイズできます。
  - `AWS_PROFILE` 環境変数または `aws.profile` システムプロパティを使用して、どのプロファイルが SDK によってロードされるかを指定できます。
4. SDK は、Amazon EC2 インスタンスメタデータサービス (IMDS) を使用して、現在実行中 Amazon EC2 のインスタンスのリージョンを特定しようとします。
  - セキュリティを強化するには、SDK が IMDS のバージョン 1 を使用しようとしないようにする必要があります。 [the section called “安全な”](#) セクションで説明されているのと同じ設定を使用して、バージョン 1 を無効にします。
5. この時点で SDK によってリージョンがまだ見つからない場合は、クライアント作成が失敗し、例外が発生します。

AWS アプリケーションを開発する場合、一般的なアプローチは、共有設定ファイル ([認証情報の取得順序で説明](#)) を使用してローカル開発用のリージョンを設定し、アプリケーションが AWS インフラストラクチャで実行されるときに、デフォルトのリージョンプロバイダーチェーンを使用してリージョンを決定することです。これによって、クライアントの作成が大幅に簡略化され、アプリケーションの性が維持されます。

## リージョンでのサービス可用性を確認する

特定の AWS のサービスがリージョンで使用できるかどうかを確認するには、サービスクライアントの `serviceMetadata` および `region` メソッドを使用します。

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

指定 AWS リージョン できる の [リージョン](#) クラスのドキュメントを参照し、サービスのエンドポイントプレフィックスを使用してクエリを実行します。

## 特定のエンドポイントを選択する

サービスのプレビュー機能を一般公開前にテストする場合など、特定の状況では、リージョン内の特定のエンドポイントを指定する必要がある場合があります。このような状況では、`endpointOverride` メソッドを呼び出してサービスクライアントを設定できます。

例えば、特定のエンドポイントで欧州 (アイルランド) リージョンを使用するように Amazon EC2 クライアントを設定するには、次のコードを使用します。

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

リージョンの現在のリストと、[すべてのサービスに対応するエンドポイントについては](#)、「リージョンとエンドポイント」を参照してください。AWS

## の SDK の起動時間を短縮する AWS Lambda

の目標の 1 AWS SDK for Java 2.x つは、関数のスタートアップレイテンシーを減らすことです AWS Lambda 。SDK には、起動時間を短縮する変更が含まれています。これについてはこのトピックの最後で説明します。

まず、このトピックでは、コールド起動時間を短縮するためにできる変更の焦点を当てます。これには、コード構造やサービスクライアントの設定の変更が含まれます。

### AWS CRT ベースの HTTP クライアントを使用する

を操作するには AWS Lambda、同期シナリオ [AwsCrtHttpClient](#) の場合は を、非同期シナリオ [AwsCrtAsyncHttpClient](#) の場合は をお勧めします。

このガイドの [the section called “AWS CRT ベースの HTTP クライアントを設定する”](#) トピックでは、HTTP クライアントを使用する利点、依存関係を追加する方法、およびサービスクライアントによるそれらの使用を設定する方法について説明します。

## 未使用の HTTP クライアント依存関係を削除する

AWS CRT ベースのクライアントの明示的な使用に加えて、SDK がデフォルトで導入する他の HTTP クライアントを削除できます。ロードする必要のあるライブラリが少ないほど、Lambda の起動時間は短くなるため、JVM がロードする必要のある未使用のアーティファクトはすべて削除する必要があります。

以下の Maven pom.xml ファイルのスニペットは、Apache ベースの HTTP クライアントと Netty ベースの HTTP クライアントが除外されていることを示しています。( AWS CRT ベースのクライアントを使用する場合、これらのクライアントは必要ありません。 ) この例では、HTTP クライアントアーティファクトを S3 クライアントの依存関係から除外し、アーaws-crt-clientアーティファクトを追加して AWS CRT ベースの HTTP クライアントへのアクセスを許可します。

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



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

### Note

pom.xml ファイル内のすべてのサービスクライアント依存関係に <exclusions> 要素を追加します。

## 検索をショートカットするようにサービスクライアントを設定します。

### リージョンを指定する

サービスクライアントを作成したら、サービスクライアントビルダーの `region` メソッドを呼び出します。これにより、SDK のデフォルトの [リージョン検索プロセス](#) がショートカットされ、複数の場所で AWS リージョン 情報を確認できます。

Lambda コードをリージョンから独立させるには、`region` メソッド内で以下のコードを使用します。このコードは、Lambda コンテナによって設定された `AWS_REGION` 環境変数にアクセスします。

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

### EnvironmentVariableCredentialProvider を使用する

リージョン情報のデフォルトの検索動作と同様に、SDK は複数の場所で認証情報を検索します。サービスクライアントを構築するときに [EnvironmentVariableCredentialProvider](#) を指定することで、SDK の検索処理にかかる時間を短縮できます。

### Note

この認証情報プロバイダーを使用すると、コードは Lambda 関数で使用できますが、Amazon EC2 やその他のシステムでは機能しない場合があります。

以下のコードスニペットは、Lambda 環境で使用するために適切に設定された S3 サービスクライアントを示しています。

```
S3Client s3Client = S3Client.builder()

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

## Lambda 関数ハンドラーの外部で SDK クライアントを初期化する

Lambda ハンドラーメソッドの外部で SDK クライアントを初期化することをお勧めします。これにより、実行コンテキストが再利用される場合、サービスクライアントの初期化をスキップできます。クライアントインスタンスとその接続を再利用することで、それ以降のハンドラーメソッドの呼び出しがより迅速に行われます。

以下の例では、S3Client インスタンスは静的ファクトリーメソッドを使用してコンストラクターで初期化されます。Lambda 環境で管理されているコンテナが再利用されると、初期化された S3Client インスタンスが再利用されます。

```
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.
    }
}
```

## 依存関係インジェクションを最小限に抑える

依存関係インジェクション (DI) フレームワークを使用すると、設定プロセスの完了までにさらに時間がかかる場合があります。また、追加の依存関係が必要になる可能性もあり、読み取りに時間がかかります。

DI フレームワークが必要な場合は、[Dagger](#) などの軽量な DI フレームワークの使用をお勧めします。

## Maven アーキタイプターゲットを使用する AWS Lambda

AWS Java SDK チームは、最小限の起動時間で Lambda プロジェクトをブートストラップするための [Maven Archetype](#) テンプレートを開発しました。アーキタイプから Maven プロジェクトを構築し、依存関係が Lambda 環境に合わせて適切に設定されていることを確認できます。

アーキタイプおよびデプロイ例について知るには、この[ブログ記事](#)を参照してください。

## Lambda SnapStart for Java を検討する

ランタイム要件に互換性がある場合、は [Lambda SnapStart for Java](#) AWS を提供します。Lambda SnapStart は、Java 関数の起動パフォーマンスを向上させるインフラストラクチャベースのソリューションです。関数の新しいバージョンを発行すると、Lambda は関数を SnapStart 初期化し、メモリとディスク状態のイミュータブルで暗号化されたスナップショットを取得します。SnapStart その後、はスナップショットをキャッシュして再利用します。

## 起動時間に影響するバージョン 2.x の変更点

コードに加えた変更に加えて、SDK for Java のバージョン 2.x には、起動時間を短縮する 3 つの主な変更が含まれています。

- 初期化時間を短縮するシリアル化ライブラリである、[Jackson-jr](#) を使用します
- JDK の一部である日付時刻オブジェクトで [java.time](#) ライブラリを使用します。
- ログ記録ファサードの [Slf4j](#) を使用します

## 追加リソース

AWS Lambda デベロッパーガイドには、Java 固有ではない Lambda 関数を開発するための[ベストプラクティスに関するセクション](#)が含まれています。

を使用するクラウドネイティブアプリケーションを Java で構築する例については AWS Lambda、この[ワークショップのコンテンツ](#)「」を参照してください。ワークショップでは、パフォーマンスの最適化やその他のベストプラクティスについて説明します。

起動時のレイテンシーを短縮するために、事前にコンパイルされた静的イメージの使用を検討できます。例えば、SDK for Java 2.x および Maven を使用して [GraalVM ネイティブイメージを構築](#) できます。

## HTTP クライアント

AWS SDK for Java 2.xで、サービスクライアントに使用する HTTP クライアントを変更するとともに、HTTP クライアントのデフォルト設定を変更できます。このセクションでは、HTTP クライアントおよび SDK を設定する方法について説明します。

### Java の SDK で使用可能な HTTP クライアント

#### 同期クライアント

SDK for Java の同期 HTTP クライアントは、[SdkHttpClient](#) インターフェイスを実装します。S3Client や DynamoDbClient などの同期サービスクライアントでは、同期 HTTP クライアントを使用する必要があります。AWS SDK for Java には 3 つの同期 HTTP クライアントがあります。

#### ApacheHttpClient (デフォルト)

[ApacheHttpClient](#) は、同期サービスクライアントのデフォルトの HTTP クライアントです。ApacheHttpClient の設定の詳細については、「[Apache ベースの HTTP クライアントの設定](#)」を参照してください。

#### AwsCrtHttpClient

[AwsCrtHttpClient](#) は、高スループットとノンブロッキング IO を提供します。AWS 共通ランタイム (CRT) Http クライアント上に構築されています。AwsCrtHttpClient の設定と、サービスクライアントと一緒に使用方法については、「[the section called “AWS CRT ベースの HTTP クライアントを設定する”](#)」を参照してください。

#### URLConnectionHttpClient

アプリケーションが使用する jar とサードパーティーライブラリの数を最小限に抑えるには、[URLConnectionHttpClient](#) を使用できます。[URLConnectionHttpClient](#) の設定の詳細については、「[URLConnection ベースの HTTP クライアントを設定する](#)」を参照してください。

## 非同期クライアント

SDK for Java の非同期 HTTP クライアントは、[SdkAsyncHttpClient](#) インターフェイスを実装します。S3AsyncClient や DynamoDbAsyncClient などの非同期サービスクライアントでは、非同期 HTTP クライアントを使用する必要があります。AWS SDK for Java には 2 つの非同期 HTTP クライアントがあります。

### NettyNioAsyncHttpClient ( デフォルト )

[NettyNioAsyncHttpClient](#) は、非同期クライアントで使用されるデフォルトの HTTP クライアントです。NettyNioAsyncHttpClient の設定の詳細については、「[the section called “Netty ベースの HTTP クライアントの設定”](#)」を参照してください。

### AwsCrtAsyncHttpClient

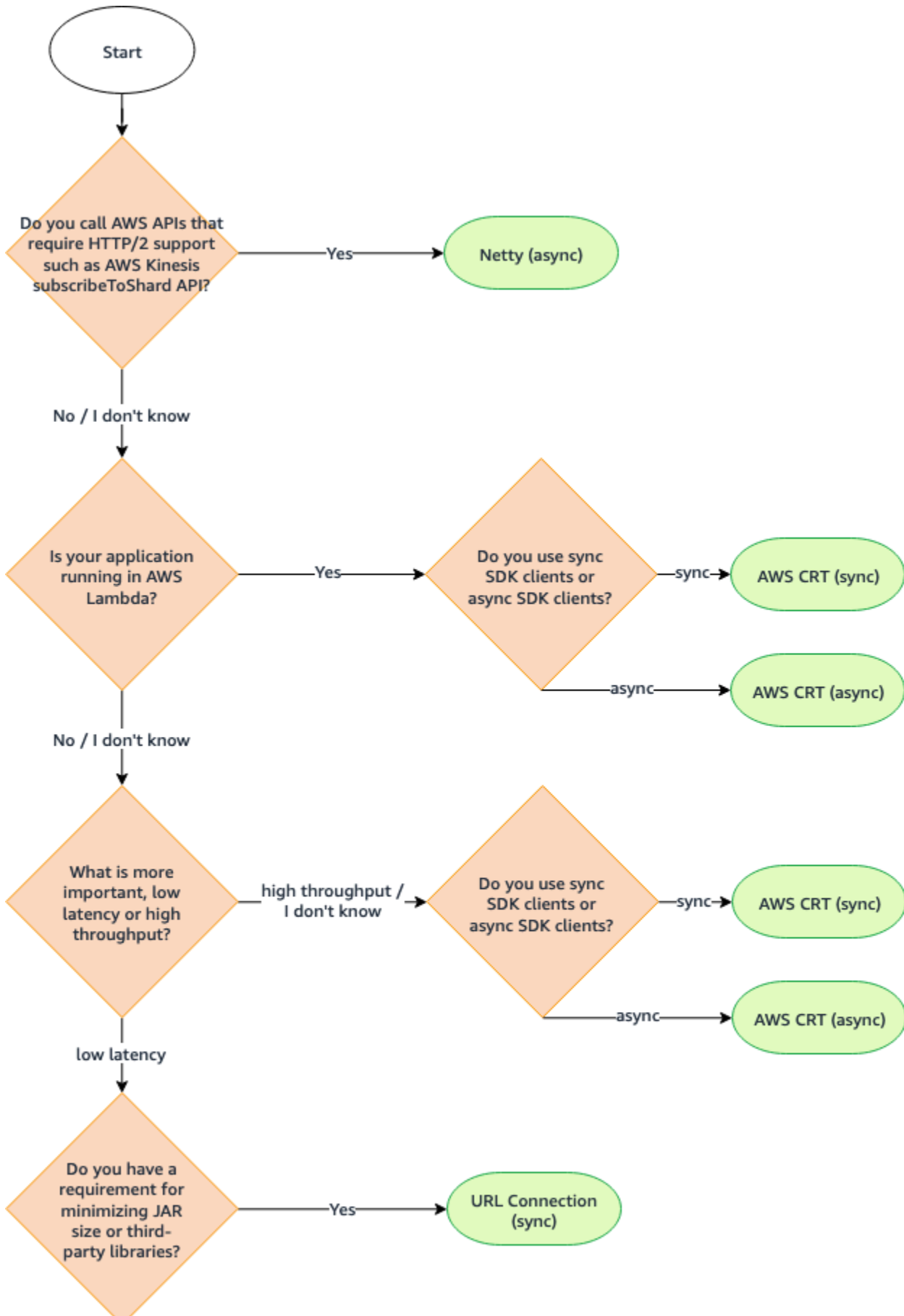
[AwsCrtAsyncHttpClient](#) は、AWS Common Runtime (CRT) HTTP Client に基づいています。AwsCrtAsyncHttpClient の設定の詳細については、「[the section called “AWS CRT ベースの HTTP クライアントを設定する”](#)」を参照してください。

## HTTP クライアントの推奨事項

HTTP クライアントの実装を選択するには、いくつかの要因が関係します。決定には以下の情報を参考にしてください。

### 推奨事項フローチャート

以下のフローチャートは、どの HTTP クライアントを使用するかを判断するのに役立つ一般的なガイダンスを示しています。



## HTTP クライアントの比較

以下の表は、各 HTTP クライアントの詳細情報を示しています。

HTTP クライアント	同期または非同期	どのようなときに使うか	制限/欠点
Apache ベースの HTTP クライアント  (デフォルトの同期 HTTP クライアント)	同期	高スループットよりも低レイテンシーを好む場合に使用します	他の HTTP クライアントと比較して起動時間が遅い
URLConnection ベースの HTTP クライアント	同期	サードパーティの依存関係を制限するという厳しい要件がある場合に使用します	Amazon ApiGateway の更新オペレーションなど一部の API で必要な HTTP PATCH メソッドはサポートしていません
AWS CRT ベースの同期 HTTP クライアント <sup>1</sup>	同期	<ul style="list-style-type: none"> <li>アプリケーションが <code>aws.crt</code> で実行されている場合に使用します。AWS Lambda</li> <li>低レイテンシーよりも高スループットを好む場合に使用します</li> <li>同期 SDK クライアントを好む場合に使用します</li> </ul>	該当なし
Netty ベースの HTTP クライアント	非同期	<ul style="list-style-type: none"> <li>アプリケーションが Kinesis APIs などの HTTP/2 サポートを必要とする API を呼び出す場合に使用します。 <a href="#">SubscribeToShard</a></li> </ul>	他の HTTP クライアントと比較して起動時間が遅い

HTTP クライアント	同期または非同期	どのようなときに使うか	制限/欠点
(デフォルトの非同期 HTTP クライアント)			
AWS CRT ベースの非同期 HTTP クライアント <sup>1</sup>	非同期	<ul style="list-style-type: none"> <li>アプリケーションが AWS Lambda で実行されている場合に使用します</li> <li>低レイテンシーよりも高スループットを好む場合に使用します</li> <li>非同期 SDK クライアントを好む場合に使用します</li> </ul>	<ul style="list-style-type: none"> <li>KinesisAsyncClient および TranscribeStreamingAsyncClient などの HTTP/2 サポートを必要とするサービスクライアントはサポートしていません</li> </ul>

<sup>1</sup>利点が追加されたため、可能であれば AWS CRT ベースの HTTP クライアントを使用することをお勧めします。

## スマート設定のデフォルト

AWS SDK for Java 2.x (バージョン 2.17.102 以降) には、スマート設定のデフォルト機能があります。この機能は、HTTP クライアントの 2 つのプロパティと HTTP クライアントに影響しない他のプロパティを最適化します。

スマート設定のデフォルトは、指定したデフォルトモードの値に基づいて、connectTimeoutInMillis および tlsNegotiationTimeoutInMillis プロパティに適切な値を設定します。デフォルトモードの値は、アプリケーションの特性に基づいて選択します。

スマート設定のデフォルトおよびアプリケーションに最適なデフォルトモードの値を選択する方法の詳細については、「[AWS SDKs and Tools リファレンスガイド](#)」を参照してください。

アプリケーションのデフォルトモードを設定する 4 つの方法は以下のとおりです。



## Service client

サービスクライアントビルダーを使用して、サービスクライアントでデフォルトモードを直接設定します。以下の例は、DynamoDbClient のデフォルトモードの出力形式を auto に設定します。

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

## System property

aws.defaultsMode システムプロパティを使用してデフォルトモードを指定できます。Java でシステムプロパティを設定する場合は、サービスクライアントを初期化する前にプロパティを設定する必要があります。

以下の例は、Java のシステムプロパティを使用して auto にデフォルトモードを設定する方法を示しています。

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

以下の例は、java コマンドの -D オプションを使用して auto にデフォルトモードを設定する方法を示しています。

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

## Environment variable

環境変数 AWS\_DEFAULTS\_MODE の値を設定して、アプリケーションのデフォルトモードを選択します。

以下の情報は、環境変数を使用して auto にデフォルトモードの値を設定するコマンドを示しています。

オペレーティングシステム	環境変数を設定するコマンド
Linux、macOS、または Unix	export AWS_DEFAULTS_MODE=auto
Windows	set AWS_DEFAULTS_MODE=auto

## AWS config file

次の例に示すように、共有 AWS config ファイルに `defaults_mode` 構成プロパティを追加できます。

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

システムプロパティ、環境変数、または AWS 設定ファイルを使用してデフォルトモードをグローバルに設定すると、HTTP クライアントを構築するときに設定を上書きできます。

`httpClientBuilder()` メソッドで HTTP クライアントを構築すると、設定は構築するインスタンスにのみ適用されます。この例を [こちら](#) に示します。この例の Netty ベースの HTTP クライアントは、`connectTimeoutInMillis` および `tlsNegotiationTimeoutInMillis` にグローバルに設定されているデフォルトモード値のをすべて上書きします。

## Apache ベースの HTTP クライアントの設定

の同期サービスクライアントは、[ApacheHttpClient](#) デフォルトで Apache ベースの HTTP クライアント AWS SDK for Java 2.x を使用します。SDK の `ApacheHttpClient` は Apache に基づいていません [HttpClient](#)。

SDK には [URLConnectionHttpClient](#) も用意されています。これは、より迅速にロードされますが、機能が少なくなります。URLConnectionHttpClient の設定の詳細については、「[the section called "URLConnection ベースの HTTP クライアントを設定する"](#)」を参照してください。

で使用できる設定オプションの完全なセットについては `ApacheHttpClient`、  
「[ApacheHttpClient.Builder](#)」および「[ProxyConfiguration.Builder](#)」を参照してください。

## ApacheHttpClient へのアクセス

ほとんどの場合、明示的な設定を行わずに `ApacheHttpClient` を使用することになります。サービスクライアントを宣言すると、SDK が標準値を使用して `ApacheHttpClient` を自動的に設定します。

`ApacheHttpClient` を明示的に設定したり、複数のサービスクライアントで使用したりする場合は、設定できるようにする必要があります。

## 設定が不要です

Maven でサービスクライアントへの依存関係を宣言すると、SDK は `apache-client` アーティファクトにランタイム依存関係を追加します。これにより、コンパイル時ではなく、ランタイム時にその `ApacheHttpClient` クラスをコードで使用できるようになります。Apache ベースの HTTP クライアントを設定していない場合は、依存関係を指定する必要はありません。

次の Maven `pom.xml` ファイルの XML スニペットでは、`<artifactId>s3</artifactId>` で宣言された依存関係によって Apache ベースの HTTP クライアントが自動的に取り込まれます。そのため、依存関係を特別に宣言する必要はありません。

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

これらの依存関係では、`ApacheHttpClient` ライブラリはランタイムのクラスパス上にのみ存在するため、明示的に HTTP 設定を変更することはできません。

## 設定が必要です

`ApacheHttpClient` を設定するには、コンパイル時間に `apache-client` ライブラリに依存関係を追加する必要があります。

次の Maven `pom.xml` ファイルの例を参照して、「`ApacheHttpClient`」を設定します。

```
<dependencyManagement>
```

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

## ApacheHttpClient を使用および設定する

サービスクライアントを構築すると同時に ApacheHttpClient のインスタンスを設定することも、複数のサービスクライアントで共有するために単一のインスタンスを設定することもできます。

いずれの方法でも、[ApacheHttpClient.Builder](#) を使用して Apache ベースの HTTP クライアントのプロパティを設定します。

### ベストプラクティス: ApacheHttpClient インスタンスをサービスクライアント専用にする

ApacheHttpClient のインスタンスを設定する必要がある場合は、専有 ApacheHttpClient インスタンスを構築することをお勧めします。これを行うには、サービスクライアントのビルダーの `httpClientBuilder` メソッドを使用します。この方法では、HTTP クライアントのライフサイクルが SDK によって管理されるため、不要になったときに ApacheHttpClient インスタンスを閉じなかった場合にメモリリークが発生するのを防ぐことができます。

次の例では、`S3Client` を作成して、`maxConnections` と `connectionTimeout` の値で ApacheHttpClient の埋め込みインスタンスを設定します。HTTP インスタンスは、`S3Client.Builder` の `httpClientBuilder` メソッドを使用して作成されます。

## インポート

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

## Code

```
S3Client s3Client = S3Client // Singleton: Use the s3Client for all requests.
    .builder()
    .httpClientBuilder(ApacheHttpClient.builder()
        .maxConnections(100)
        .connectionTimeout(Duration.ofSeconds(5))
    ).build();

// Perform work with the s3Client.

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

## 別の方法: **ApacheHttpClient** インスタンスを共有する

アプリケーションのリソースとメモリの使用量を低く抑えるため、**ApacheHttpClient** を設定して複数のサービスクライアント間で共有できます。HTTP 接続プールは共有されるため、リソースの使用量が少なくなります。

### Note

**ApacheHttpClient** インスタンスを共有する場合、破棄の準備が整った時点でインスタンスを閉じる必要があります。サービスクライアントが閉じられても、SDK はインスタンスを閉じません。

次の例では、2つのサービスクライアントで使用される Apache ベースの HTTP クライアントを設定します。設定された **ApacheHttpClient** インスタンスは、各ビルダーの `httpClient` メソッドに渡されます。サービスクライアントと HTTP クライアントが不要になると、コードによって明示的に閉じられます。HTTP クライアントは最後に閉じられます。

## インポート

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

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.s3.S3Client;
```

## Code

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

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

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

// Perform work with the s3Client and dynamoDbClient.

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

## プロキシの設定例

次のコードスニペットは、[Apache HTTP クライアントのプロキシ設定ビルダー](#)を使用しています。

```
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();
```

プロキシ設定に対応する Java システムプロパティは、以下のコマンドラインスニペットに示されています。

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

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

環境変数を使用する同等の設定は次のとおりです。

```
// 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 クライアントは現在、HTTPS プロキシシステムプロパティまたは HTTPS\_PROXY 環境変数をサポートしていません。

## URLConnection ベースの HTTP クライアントを設定する

は、デフォルトの [URLConnectionHttpClient](#) と比較して軽量な HTTP クライアント AWS SDK for Java 2.x を提供します ApacheHttpClient。URLConnectionHttpClient は Java [URLConnection](#) をベースにしています。

URLConnectionHttpClient は Apache ベースの HTTP クライアントよりも読み取りが速くなりますが、機能は限られています。読み取りが速いため、Java AWS Lambda 関数に [適したソリューション](#) です。

URLConnectionHttpClient には、アクセスができる [設定可能なオプション](#) がいくつかあります。

#### Note

URLConnectionHttpClient は HTTP PATCH メソッドをサポートしていません。

少数の AWS API オペレーションには PATCH リクエストが必要です。これらのオペレーション名は通常、Update\* で始まります。次にいくつかの例を示します。

- AWS Security Hub API の [いくつかのUpdate\\*オペレーション](#) と [BatchUpdateFindings](#) オペレーション
- すべての Amazon API Gateway API [Update\\* オペレーション](#)
- Amazon WorkDocs API の [いくつかのUpdate\\*オペレーション](#)

を使用する可能性がある場合は `URLConnectionHttpClient`、まず使用しているの API AWS のサービス リファレンスを参照してください。必要なオペレーションに PATCH オペレーションが使用されているかどうかを確認してください。

## `URLConnectionHttpClient` へのアクセス

`URLConnectionHttpClient` を設定して使用するには、`pom.xml` ファイル内の `url-connection-client` Maven アーティファクトへの依存関係を宣言します。

`ApacheHttpClient` とは異なり、`URLConnectionHttpClient` はプロジェクトに自動的に追加されないため、明示的に宣言する必要があります。

次の `pom.xml` ファイル例は、HTTP クライアントの使用と設定に必要な依存関係を示しています。

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

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

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



```
<artifactId>url-connection-client</artifactId>
</dependency>
</dependencies>
```

## URLConnectionHttpClient を使用および設定する

サービスクライアントを構築すると同時に `URLConnectionHttpClient` のインスタンスを設定することも、単一インスタンスを複数のサービスクライアント間で共有するように設定することもできます。

どちらの方法でも、[URLConnectionHttpClient.Builder](#) を使用して `URLConnection` ベースの HTTP クライアントのプロパティを設定します。

ベストプラクティス: `URLConnectionHttpClient` インスタンスをサービスクライアント専有にする

`URLConnectionHttpClient` のインスタンスを設定する必要がある場合は、専有 `URLConnectionHttpClient` インスタンスを構築することをお勧めします。これを行うには、サービスクライアントのビルダーの `httpClientBuilder` メソッドを使用します。この方法では、HTTP クライアントのライフサイクルが SDK によって管理されるため、不要になったときに `URLConnectionHttpClient` インスタンスを閉じなかった場合にメモリリークが発生するのを防ぐことができます。

次の例では、`S3Client` を作成して、`socketTimeout` と `proxyConfiguration` の値で `URLConnectionHttpClient` の埋め込みインスタンスを設定します。`proxyConfiguration` メソッドは、タイプ `Consumer<ProxyConfiguration.Builder>` の Java Lambda 式を取ります。

インポート

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

Code

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

```
        .socketTimeout(Duration.ofMinutes(5))
        .proxyConfiguration(proxy -> proxy.endpoint(URI.create("http://
proxy.mydomain.net:8888"))))
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();

// Perform work with the s3Client.

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

## 別の方法: `URLConnectionHttpClient` インスタンスを共有する

アプリケーションのリソースとメモリの使用量を低く抑えるため、`URLConnectionHttpClient` を設定して複数のサービスクライアント間で共有できます。HTTP 接続プールは共有されるため、リソースの使用量が少なくなります。

### Note

`URLConnectionHttpClient` インスタンスを共有する場合、破棄の準備が整った時点でインスタンスを閉じる必要があります。サービスクライアントが閉じられても、SDK はインスタンスを閉じません。

次の例では、2つのサービスクライアントが使用する `URLConnection` ベースの HTTP クライアントを設定します。設定された `URLConnectionHttpClient` インスタンスは、各ビルダーの `httpClient` メソッドに渡されます。サービスクライアントと HTTP クライアントが不要になると、コードによって明示的に閉じられます。HTTP クライアントは最後に閉じられます。

## インポート

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

## Code

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

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

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

// Perform work with the s3Client and dynamoDbClient.

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

## URLConnectionHttpClient と ApacheHttpClient を同時に使用する

アプリケーション内で `URLConnectionHttpClient` を使用するとき、各サービスクライアントに `URLConnectionHttpClient` インスタンス、またはサービスクライアントビルダーの `httpClientBuilder` メソッドを使用して `ApacheHttpClient` インスタンスを提供する必要があります。

プログラムが複数のサービスクライアントを使用していて、次の両方に該当すると、例外が発生します。

- 1つのサービスクライアントが `URLConnectionHttpClient` のインスタンスを使用するように設定されています。
- 別のサービスクライアントは、`httpClient()` または `httpClientBuilder()` メソッドで明示的に構築せずにデフォルト `ApacheHttpClient` を使用します。

例外は、クラスパス上に複数の HTTP 実装が見つかったことを示します。

次のコードスニペット例では例外が発生します。

```
// 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();
```

S3Client を ApacheHttpClient で明示的に設定することで、例外を回避できます。

```
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

ApacheHttpClient を明示的に作成するには、Maven プロジェクトファイル内の `apache-client` アーティファクトへの [依存関係を追加する](#) 必要があります。

## プロキシ設定例

次のコードスニペットでは、[URL 接続 HTTP クライアントのプロキシ設定ビルダー](#)を使用しています。

```
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();
```

プロキシ設定に対応する Java システムプロパティは、以下のコマンドラインスニペットに示されています。

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

環境変数を使用する同等の設定は次のとおりです。

```
// 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**

URLConnection ベースの HTTP クライアントは現在、HTTPS プロキシシステムプロパティまたは HTTPS\_PROXY 環境変数をサポートしていません。

## Netty ベースの HTTP クライアントの設定

の非同期オペレーションのデフォルトの HTTP クライアント AWS SDK for Java 2.x は Netty ベースの [NettyNioAsyncHttpClient](#) です。Netty ベースのクライアントは、[Netty プロジェクト](#)の非同期イベント駆動型ネットワークフレームワークに基づいています。

別の HTTP クライアントとして、新しい [AWS CRT ベースの HTTP クライアント](#)を使用することもできます。このトピックでは、NettyNioAsyncHttpClient を設定する方法を示します。

### NettyNioAsyncHttpClient へのアクセス

ほとんどの場合、非同期プログラムでは明示的な設定を行わずに NettyNioAsyncHttpClient を使用します。非同期サービスクライアントを宣言すると、SDK が標準値を使用して NettyNioAsyncHttpClient を自動的に設定します。

NettyNioAsyncHttpClient を明示的に設定したり、複数のサービスクライアントで使用したりする場合は、設定できるようにする必要があります。

#### 設定が不要です

Maven でサービスクライアントへの依存関係を宣言すると、SDK は netty-nio-client アーティファクトにランタイム依存関係を追加します。これにより、コンパイル時ではなく、ランタイム時にその NettyNioAsyncHttpClient クラスをコードで使用できるようになります。Netty ベースの HTTP クライアントを設定しない場合は、そのクライアントへの依存関係を指定する必要はありません。

次の Maven pom.xml ファイルの XML スニペットでは、<artifactId>dynamodb-enhanced</artifactId> で宣言された依存関係によって Netty ベースの HTTP クライアントが推移的に取り込まれます。そのために依存関係を特別に宣言する必要はありません。

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

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

これらの依存関係では、NettyNioAsyncHttpClient ライブラリランタイムクラスパス上にのみ存在するため、HTTP 設定を変更することはできません。

### 設定が必要です

NettyNioAsyncHttpClient を設定するには、コンパイル時に netty-nio-client アーティファクトへの依存関係を追加する必要があります。

次の Maven pom.xml ファイルの例を参照して、「NettyNioAsyncHttpClient」を設定します。

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

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

## NettyNioAsyncHttpClient を使用および設定する

サービスクライアントを構築すると同時に NettyNioAsyncHttpClient のインスタンスを設定することも、単一インスタンスを複数のサービスクライアント間で共有するように設定することもできます。

どちらの方法でも、[NettyNioAsyncHttpClient.Builder](#) を使用して Netty ベースの HTTP クライアントインスタンスのプロパティを設定します。

ベストプラクティス: **NettyNioAsyncHttpClient** インスタンスをサービスクライアント専用にする

NettyNioAsyncHttpClient のインスタンスを設定する必要がある場合は、専有 NettyNioAsyncHttpClient インスタンスを構築することをお勧めします。これを行うには、サービスクライアントのビルダーの `httpClientBuilder` メソッドを使用します。この方法では、HTTP クライアントのライフサイクルが SDK によって管理されるため、不要になったときに NettyNioAsyncHttpClient インスタンスを閉じなかった場合にメモリリークが発生するのを防ぐことができます。

次の例では、`DynamoDbEnhancedAsyncClient` インスタンスが使用する `DynamoDbAsyncClient` インスタンスを作成します。 `DynamoDbAsyncClient` インスタンスには、`connectionTimeout` と `maxConcurrency` 値を持つ `NettyNioAsyncHttpClient` インスタンスが含まれています。HTTP インスタンスは `DynamoDbAsyncClient.Builder` の `httpClientBuilder` メソッドを使用して作成されます。

### インポート

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



```
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.time.Duration;
```

## Code

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

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

// Perform work with the dynamoDbAsyncClient and enhancedClient.

// Requests completed: Close dynamoDbAsyncClient.
dynamoDbAsyncClient.close();
```

## 代替アプローチ: **NettyNioAsyncHttpClient** インスタンスを共有する

アプリケーションのリソースとメモリの使用量を低く抑えるため、**NettyNioAsyncHttpClient** を構成して複数のサービスクライアント間で共有できます。HTTP 接続プールは共有されるため、リソースの使用量が少なくなります。

### Note

**NettyNioAsyncHttpClient** インスタンスを共有する場合、それを破棄する準備が整った時点でインスタンスを閉じる必要があります。サービスクライアントが閉じられても、SDK はインスタンスを閉じません。

次の例では、2つのサービスクライアントが使用する Netty ベースの HTTP クライアントを設定します。設定された `NettyNioAsyncHttpClient` インスタンスは、各ビルダーの `httpClient` メソッドに渡されます。サービスクライアントと HTTP クライアントが不要になると、コードによって明示的に閉じられます。HTTP クライアントは最後に閉じられます。

## インポート

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

## Code

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

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

DynamoDbAsyncClient dbAsyncClient =
    DynamoDbAsyncClient.builder()
        .httpClient(nettyHttpClient)
        .defaultsMode(DefaultsMode.IN_REGION)

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

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

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

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

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

```
s3AsyncClient.close();
dbAsyncClient.close()
nettyHttpClient.close(); // Explicitly close nettyHttpClient.
```

## プロキシの設定例

次のコードスニペットは [Netty HTTP クライアントのプロキシ設定ビルダー](#) を使用しています。

```
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();
```

プロキシ設定に対応する Java システムプロパティは、以下のコマンドラインスニペットに示されています。

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

### Important

HTTPS プロキシのシステムプロパティのいずれかを使用するには、scheme プロパティをコード内で https に設定する必要があります。スキームプロパティがコードで設定されていない場合、スキームのデフォルトは HTTP になり、SDK は http.\* システムプロパティのみを検索します。

環境変数を使用する同等の設定は次のとおりです。

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

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

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

## AWS CRT ベースの HTTP クライアントを設定する

AWS CRT ベースの HTTP クライアントには、同期 [AwsCrhttpClient](#) および非同期 が含まれま  
す [AwsCrhttpClient](#)。AWS CRT ベースの HTTP クライアントには、次の HTTP クライアント  
の利点があります。

- SDK 起動時間の短縮
- より小さなメモリフットプリント
- レイテンシータイムの短縮
- 接続のヘルス管理
- DNS ロードバランサー

### AWS SDK の CRT ベースのコンポーネント

このトピックで説明する AWS CRT ベースの HTTP クライアントと AWS CRT ベースの S3 クライ  
アントは、SDK のさまざまなコンポーネントです。

同期および非同期の AWS CRT ベースの HTTP クライアントは、SDK HTTP インターフェイスを実  
装したもので、一般的な HTTP 通信に使用されます。これらは SDK の他の同期 または 非同期の  
HTTP クライアントに代わるものですが、その他の利点もあります。

[AWS CRT ベースの S3 クライアント](#) は [S3AsyncClient](#) インターフェイスの実装であり、Amazon S3  
サービスの使用に使用されます。これは [S3AsyncClient](#) インターフェイスの Java ベース実装に  
代わるもので、いくつかの利点があります。

どちらのコンポーネントも [AWS Common Runtime](#) のライブラリを使用していますが、AWS CRT  
ベースの HTTP クライアントは [aws-c-s3 ライブラリ](#) を使用せず、[S3 マルチパートアップロー  
ド API](#) 機能はサポートしていません。これとは対照的に、AWS CRT ベースの S3 クライアント  
は、S3 マルチパートアップロード API 機能をサポートすることを目的として構築されました。

## AWS CRT ベースの HTTP クライアントにアクセスする

AWS CRT ベースの HTTP クライアントを使用する前に、最小バージョン 2.22.0 の `aws-crt-client` アーティファクトをプロジェクトの依存関係に追加します。

次の Maven は、部品表 (BOM) メカニズムを使用して宣言された AWS CRT ベースの HTTP クライアント `pom.xml` を示しています。

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

Maven central リポジトリにアクセスして、[最新バージョン](#)を確認します。

## AWS CRT ベースの HTTP クライアントの使用と設定

AWS CRT ベースの HTTP クライアントは、サービスクライアントの構築とともに設定することも、複数のサービスクライアント間で共有するように 1 つのインスタンスを設定することもできます。

どちらの方法でも、ビルダーを使用して AWS CRT ベースの HTTP クライアントインスタンスの [プロパティを設定します](#)。

## ベストプラクティス: インスタンスをサービスクライアント専有にする

AWS CRT ベースの HTTP クライアントのインスタンスを設定する必要がある場合は、サービスクライアントとともに構築して、インスタンスを専用にするをお勧めします。これを行うには、サービスクライアントのビルダーの `httpClientBuilder` メソッドを使用します。このようにして、HTTP クライアントのライフサイクルは SDK によって管理されるため、AWS CRT ベースの HTTP クライアントインスタンスが不要になったときにシャットダウンされない場合にメモリリークが発生する可能性を回避できます。

次の例では、S3 サービスクライアントを作成し、`connectionTimeout` および `maxConcurrency` 値を使用して AWS CRT ベースの HTTP クライアントを設定します。

### Synchronous client

#### インポート

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

#### Code

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

// Perform work with the s3Client.

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

### Asynchronous client

#### インポート

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

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

## Code

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

// Perform work with the s3AsyncClient.

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

## 別の方法: インスタンスを共有する

アプリケーションのリソースとメモリの使用量を減らすために、AWS CRT ベースの HTTP クライアントを設定し、複数のサービスクライアント間で共有できます。HTTP 接続プールは共有されるため、リソースの使用量が少なくなります。

### Note

AWS CRT ベースの HTTP クライアントインスタンスを共有する場合は、破棄の準備ができたら閉じます。サービスクライアントが閉じられても、SDK はインスタンスを閉じません。

次の例では、`connectionTimeout`および`maxConcurrency`値を使用して AWS CRT ベースの HTTP クライアントインスタンスを設定します。設定された `httpClient` インスタンスは、各サービスクライアントのビルダーのメソッドに渡されます。サービスクライアントと HTTP クライアントが不要になると、それらは明示的に閉じられます。HTTP クライアントは最後に閉じられます。

## Synchronous client

### インポート

```
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
```

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

## Code

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

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

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

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

## Asynchronous client

### インポート

```
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.awscore.defaultsmode.DefaultsMode;
```



```
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.crt.AwsCrtAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import java.time.Duration;
```

## Code

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

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

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

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

## AWS CRT ベースの HTTP クライアントをデフォルトとして設定する

Maven ビルドファイルは、SDK が AWS CRT ベースの HTTP クライアントをサービスクライアントのデフォルトの HTTP クライアントとして使用するよう設定できます。

そのためには、デフォルトの HTTP クライアント依存関係を持つ `exclusions` 要素を各サービスクライアントアーティファクトに追加します。

次のpom.xml例では、SDKはS3サービスにAWS CRTベースのHTTPクライアントを使用します。コード内のサービスクライアントがS3AsyncClientの場合、SDKはAwsCrtAsyncHttpClientを使用します。サービスクライアントがS3Clientの場合、SDKはAwsCrtHttpClientを使用します。この設定では、デフォルトのNettyベースの非同期HTTPクライアントと、デフォルトのApacheベースの同期HTTPは使用できません。

```
<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>
```

Maven central リポジトリにアクセスして、最新の####値を確認します。

#### Note

1つのpom.xmlファイル内で複数のサービスクライアントが宣言されている場合は、すべてexclusions XML要素を必要とします。

## Java システムプロパティを使用する

AWS CRT ベースの HTTP クライアントをアプリケーションのデフォルトの HTTP として使用するには、Java システムプロパティの値 `software.amazon.awssdk.http.async.service.impl` に設定できず `software.amazon.awssdk.http.crt.AwsCrtSdkHttpClient`。

アプリケーションの起動時に設定するには、次のようなコマンドを実行します。

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

次のコードスニペットを使用して、アプリケーションコードでシステムプロパティを設定します。

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

### Note

システムプロパティを使用して AWS CRT ベースの HTTP クライアントの使用を設定する場合は、`pom.xml` ファイル内の `aws-crt-client` アーティファクトに依存関係を追加する必要があります。

## AWS CRT ベースの HTTP クライアントの詳細設定

AWS CRT ベースの HTTP クライアントのさまざまな設定を使用できます。これには、接続のヘルス設定や最大アイドル時間が含まれます。 `AwsCrtAsyncHttpClient` で [利用可能な設定オプション](#)を確認できます。 `AwsCrtHttpClient` にも同じオプションを設定できます。

### 接続のヘルス設定

HTTP クライアントビルダーの `connectionHealthConfiguration` メソッドを使用して、AWS CRT ベースの HTTP クライアントの接続ヘルス設定を構成できます。

次の例では、接続ヘルス設定と接続の最大アイドル時間で設定された AWS CRT ベースの HTTP クライアントインスタンスを使用する S3 サービスクライアントを作成します。

### Synchronous client

#### インポート

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

## Code

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

// Perform work with s3Client.

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

## Asynchronous client

### インポート

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

## Code

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

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

## HTTP/2 サポート

HTTP/2 プロトコルは AWS CRT ベースの HTTP クライアントではまだサポートされていませんが、将来のリリースが予定されています。

その間に、[KinesisAsyncClient](#)や などの HTTP/2 サポートを必要とするサービスクライアントを使用している場合は[TranscribeStreamingAsyncClient](#)、[NettyNioAsyncHttpClient](#)代わりに を使用することを検討してください。

## プロキシ設定例

次のコードスニペットは、コード内でプロキシの設定を行うために使用する [ProxyConfiguration.Builder](#) の使用方法を示しています。

### Synchronous client

#### インポート

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

#### Code

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

## Asynchronous client

### インポート

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

### Code

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

プロキシ設定に対応する Java システムプロパティは、以下のコマンドラインスニペットに示されています。

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

### Important

HTTPS プロキシのシステムプロパティのいずれかを使用するには、scheme プロパティをコード内で https に設定する必要があります。スキームプロパティがコードで設定されていない場合、スキームのデフォルトは HTTP になり、SDK は http.\* システムプロパティのみを検索します。

環境変数を使用する同等の設定は次のとおりです。

```
// Set the following environment variables.
```

```
// $ export HTTPS_PROXY="https://username:password@myproxy:1234"  
// $ export NO_PROXY="localhost|host.example.com"  
  
// Set the 'useSystemPropertyValues' to false on the proxy configuration.  
SdkAsyncHttpClient crtAsyncHttpClient = AwsCrtAsyncHttpClient.builder()  
    .proxyConfiguration(ProxyConfiguration.builder()  
        .scheme("https")  
        .useSystemPropertyValues(Boolean.FALSE)  
        .build())  
    .build();  
  
// Run the application.  
// $ java -cp ... App
```

## HTTP プロキシの設定

HTTP プロキシを設定するには、コードを使用するか、Java システムプロパティを設定するか、環境変数を設定します。

### コードで を設定する

サービスクライアントをビルドするときに、クライアント固有の ProxyConfiguration ビルダーを使用してコードでプロキシを設定します。次のコードは、Amazon S3 サービスクライアントで使用される Apache ベースの HTTP クライアントのプロキシ設定の例を示しています。

```
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();
```

このトピックの各 HTTP クライアントのセクションでは、プロキシ設定の例を示しています。

- [Apache HTTP クライアント](#)
- [URLConnection ベースの HTTP クライアント](#)

- [Netty ベースの HTTP クライアント](#)
- [AWS CRT ベースの HTTP クライアント](#)

## 外部設定で HTTP プロキシを設定する

コードでProxyConfigurationビルダーを明示的に使用していない場合でも、SDK はデフォルトのプロキシ設定を構成するための外部設定を探します。

デフォルトでは、SDK はまず JVM システムプロパティを検索します。プロパティが 1 つでも見つかった場合、SDK は 値とその他のシステムプロパティ値を使用します。使用可能なシステムプロパティがない場合、SDK はプロキシ環境変数を探します。

SDK では、次の Java システムプロパティと環境変数を使用できます。

### Java のシステムプロパティ

システムプロパティ	説明	HTTP クライアントのサポート
HTTP.プロキシホスト	HTTP プロキシサーバーのホスト名	すべて
http.proxyPort	HTTP プロキシサーバーのポート番号	すべて
http.proxyUser	HTTP プロキシ認証用のユーザー名	すべて
http.proxyPassword	HTTP プロキシ認証用のパスワード	すべて
http.nonProxyHosts	プロキシをバイパスして直接アクセスする必要があるホストのリスト。 <a href="#">このリストは、HTTPS を使用する場合にも有効です。</a>	すべて
https.proxyHost	HTTPS プロキシサーバーのホスト名	Netty、CRT



システムプロパティ	説明	HTTP クライアントのサポート
https.proxyPort	HTTPS プロキシサーバーのポート番号	Netty、CRT
https.proxyUser	HTTPS プロキシ認証用のユーザー名	Netty、CRT
https.proxyPassword	HTTPS プロキシ認証用のパスワード	Netty、CRT

## 環境変数

環境変数	説明	HTTP クライアントのサポート
HTTP_PROXY <sup>1</sup>	HTTP のスキームを持つ有効な URL	すべて
HTTPS_PROXY <sup>1</sup>	HTTPS のスキームを持つ有効な URL	Netty、CRT
NO_PROXY <sup>2</sup>	プロキシをバイパスして直接アクセスする必要があるホストのリスト。リストは HTTP と HTTPS の両方で有効です。	すべて

## キーとフライトノートを表示する

すべて - SDK が提供するすべての HTTP クライアント —

URLConnectionHttpClient、ApacheHttpClient、NettyNioAsyncHttpClient、AwsCrtAsyncHttp

Netty - Netty ベースの HTTP クライアント (NettyNioAsyncHttpClient)。

CRT - AWS CRT ベースの HTTP クライアント (AwsCrtHttpClient および AwsCrtAsyncHttpClient)。

<sup>1</sup>クエリされる環境変数は、HTTP\_PROXYまたはHTTPS\_PROXY、クライアントのスキーム設定によって異なりますProxyConfiguration。デフォルトのスキームはHTTPです。次のスニペットは、環境変数の解決に使用されるHTTPSにスキームを変更する方法を示しています。

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

<sup>2</sup>NO\_PROXY環境変数は、ホスト名間の「|」と「」の区切り文字の組み合わせをサポートします。ホスト名には「\*」ワイルドカードを含めることができます。

## 設定の組み合わせを使用する

コード、システムプロパティ、環境変数でHTTPプロキシ設定を組み合わせで使用できます。

### Example – システムプロパティとコードによって提供される設定

```
// 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 は次のプロキシ設定を解決します。

```
Host = localhost
```

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

### Example — システムプロパティと環境変数の両方が使用可能

各 HTTP クライアントのProxyConfigurationビルダーは、useSystemPropertyValuesおよびという名前の設定を提供しますuseEnvironmentVariablesValues。デフォルトでは、両方の設定はtrueです。trueの場合、SDKはProxyConfigurationビルダーによって提供されていないオプションに、システムプロパティまたは環境変数の値を自動的に使用します。

#### Important

システムプロパティは、環境変数よりも優先されます。HTTP プロキシシステムプロパティが見つかった場合、SDKはシステムプロパティからすべての値を取得し、環境変数からは取得しません。システムプロパティよりも環境変数に優先順位を付ける場合は、useSystemPropertyValuesをfalseに設定します。

この例では、次の設定がランタイムとして使用できます。

```
// 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"
```

サービスクライアントは、次のいずれかのステートメントを使用して作成されます。どのステートメントもプロキシ設定を明示的に設定していません。

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

```
.build())  
.build();
```

次のプロキシ設定は SDK によって解決されます。

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

サービスクライアントにはデフォルトのプロキシ設定があるため、SDK はシステムプロパティを検索し、次に環境変数を検索します。システムプロパティ設定は環境変数よりも優先されるため、SDK はシステムプロパティのみを使用します。

次のコード `false` に示すように、システムプロパティの使用が `true` に変更された場合、SDK は環境変数のみを解決します。

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

HTTP を使用して解決されたプロキシ設定は次のとおりです。

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

## AWS SDK for Java 2.x の例外処理

AWS SDK for Java 2.x によって例外がスローされる仕組みとタイミングを理解することは、SDK を使用して高品質なアプリケーションを構築するために重要なことです。以下のセクションでは、SDK によってスローされる例外のさまざまなケース、および例外の適切な処理方法について説明します。

## 非チェック例外を使用する理由

AWS SDK for Java では、次の理由で、チェック例外ではなく実行時例外 (非チェック例外) が使用されます。

- 懸念がない場合には例外ケースの処理を強制的に適用せずに (さらにコードを冗長にすることなく)、開発者がエラーを細かく制御できるようにするため。
- サイズの大きいアプリケーションで、チェック例外に関連する拡張性の問題が発生するのを防ぐため。

一般的に、チェック例外は小規模なアプリケーションでは役立ちますが、アプリケーションのサイズが大きくなり、複雑化すると、チェック例外が問題となる場合があります。

### AwsServiceException ( およびサブクラス )

[AwsServiceException](#) は、の使用時に発生する最も一般的な例外ですAWS SDK for Java。

AwsServiceExceptionは、より一般的な のサブクラスです[SdkServiceException](#)。

AwsServiceExceptionは、からのエラーレスポンスを表しますAWS のサービス。例えば、存在しない Amazon EC2 インスタンスを終了しようとする、Amazon EC2 からエラー応答が返り、そのエラー応答の詳細はすべて、スローされる AwsServiceException に含まれます。

AwsServiceException が発生した場合、リクエストは AWS のサービス に適切に送信されましたが、正常に処理できませんでした。これは、リクエストのパラメータに含まれるエラーまたはサービス側の問題が原因です。

AwsServiceException では次のような情報がわかります。

- 返された HTTP ステータスコード
- 返された AWS エラーコード
- [AwsErrorDetails](#) クラス内のサービスからの詳細なエラーメッセージ
- 失敗したリクエストの AWS リクエスト ID

ほとんどの場合、AwsServiceException のサービス固有のサブクラスがスローされ、開発者はキャッチブロックを通じてエラーケースの処理を細かく制御できます。の Java SDK API リファレンスには、多数のAwsServiceExceptionサブクラス[AwsServiceException](#)が表示されます。サブクラスリンクを使用してドリルダウンすると、サービスによってスローされた詳細な例外を確認できます。

たとえば、以下の SDK API リファレンスへのリンクは、いくつかの一般的なAWS のサービスの例外階層を示しています。各ページに表示されるサブクラスのリストには、コードがキャッチできる特定の例外が表示されます。

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

例外の詳細については、[AwsErrorDetails](#) オブジェクト `errorCode` の を調べます。 `errorCode` 値を使用して、サービスガイド API の情報を検索できます。たとえば、 `S3Exception` がキャッチされ、 `AwsErrorDetails#errorCode()` 値が `InvalidRequest` の場合、Amazon S3 API リファレンスの [エラーコードのリスト](#) を使用して詳細を確認してください。

## SdkClientException

[SdkClientException](#) は、 にリクエストを送信しようとしたとき、AWSまたはからのレスポンスを解析しようとしたときに、Java クライアントコード内で問題が発生したことを示します AWS。 `SdkClientException` は、一般的に `SdkServiceException` よりも深刻な例外で、クライアントが AWS のサービスに対するサービス呼び出しを実行できないという重大な問題を示しています。たとえば、いずれかのクライアントでオペレーションを呼び出そうとしたときに、ネットワーク接続が利用できない場合、AWS SDK for Java は `SdkClientException` をスローします。

## 例外と再試行動作

SDK for Java は、いくつかの [クライアント側の例外と](#)、AWS のサービス レスポンスから受け取った [HTTP ステータスコード](#) のリクエストを再試行します。これらのエラーは、サービスクライアントがデフォルトで使用するレガシー `RetryMode` の一部として処理されます。 [RetryMode](#) の Java API リファレンスには、モードを設定するさまざまな方法が記載されています。

自動再試行をトリガーする例外と HTTP ステータスコードをカスタマイズするには、 [RetryOnExceptionsCondition](#)、 [RetryOnStatusCodeCondition](#) インスタンスを追加する [RetryPolicy](#) をサービスクライアントに設定します。

## SDK for Java 2.x でのログ記録

は [SLF4J](#) AWS SDK for Java 2.x を使用します。これは、実行時に複数のログ記録システムのいずれかを使用できるようにする抽象化レイヤーです。

サポートされるログ記録システムには、Java ログ記録 フレームワークおよび Apache [Log4j 2](#) などがあります。このトピックでは、SDK を使用するためのログ記録システムとして Log4j 2 を使用する方法について説明します。

## Log4j 2 設定ファイル

通常は `log4j2.xml` Log4j 2 という名前の設定ファイルを使用します。設定ファイルの例を次に示します。この設定ファイルで使用する値の詳細については、「[Log4j 設定のマニュアル](#)」を参照してください。

`log4j2.xml` ファイルは、アプリケーションの起動時にクラスパス上にある必要があります。Maven プロジェクトの場合は、ファイルを `<project-dir>/src/main/resources` ディレクトリに置きます。

`log4j2.xml` 設定ファイルは、[ログ記録レベル](#)、ログ記録出力の送信先 ([ファイルまたはコンソールなど](#))、[出力フォーマット](#)などのプロパティを指定します。ログ記録レベルは Log4j 2 が出力する詳細レベルを指定します。Log4j 2 では、複数のログ記録[階層](#)の概念をサポートしています。ログ記録レベルは、階層ごとに個別に設定されます。で使用する主なログ記録階層 AWS SDK for Java 2.x は `software.amazon.awssdk` です。

## ログ記録を追加する

構築ファイルで SLF4J の Log4j 2 バインディングを設定するには、以下を使用します。

### Maven

次の要素を `pom.xml` ファイルに追加します。

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

### Gradle–Kotlin DSL

次のコードを `build.gradle.kts` ファイルに追加します。

```
...
dependencies {
```

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

log4j-slf4j2-impl アーティファクトの最小バージョンには 2.20.0 を使用します。最新バージョンには、[Maven Central](#) に公開されているバージョンを使用します。VERSION は、使用するバージョンに置き換えます。

## SDK 固有のエラーおよび警告

SDK のクライアントライブラリからの重要なメッセージを取得するために、

「software.amazon.awssdk」ロガー階層は必ず「WARN」に設定しておくことをお勧めします。例えば、アプリケーションで InputStream が正しく閉じられなかったためにリソースが漏洩する可能性がある場合、Amazon S3 クライアントが検出した場合、S3 クライアントは警告メッセージを使用してログに報告します。これにより、リクエストやレスポンスの処理でクライアントに問題が発生した場合、メッセージが必ずログに記録されます。

以下の log4j2.xml ファイルでは、rootLogger を「WARN」に設定しています。これにより、「software.amazon.awssdk」階層内のロガーを含め、アプリケーション内のすべてのロガーからの警告およびエラーレベルのメッセージが出力されます。また、<Root level="ERROR"> が使用されている場合、明示的に「software.amazon.awssdk」ロガー階層を「WARN」に設定することもできます。

### Log4j2.xml 設定ファイルの例

この設定では、すべてのロガー階層の「ERROR」レベルと「WARN」レベルのメッセージがコンソールにログ記録されます。

```
<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>
```



```
</Root>
</Loggers>
</Configuration>
```

## リクエスト/応答の概要のログ記録

へのリクエストごとに一意の AWS リクエスト ID AWS のサービス が生成されます。これは、AWS のサービス がリクエストを処理する方法に問題が発生した場合に役立ちます。AWS リクエスト IDs 「software.amazon.awssdk.request」 ロガーの「DEBUG」 ログレベルを介して報告することもできます。 [SdkServiceException](#)

次の log4j2.xml ファイルでは、リクエストとレスポンスの要約が有効になっています。

```
<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>
```

ログ出力の例を次に示します。

```
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:
QS9DUMME2NHEDH8TGT9N5V530JVV4KQNS05AEMVJF66Q9ASUAAJG, Extended Request ID: not
available
```

リクエスト ID だけに興味がある場合は `<Logger name="software.amazon.awssdk.requestId" level="DEBUG" />` を使用します。

## デバッグレベルの SDK ログ記録

SDK の動作についてさらに詳しく知りたい場合は、`software.amazon.awssdk` ロガーのログ記録レベルを `DEBUG` に設定できます。このレベルでは、SDK は大量の詳細を出力するため、統合テストを使用してエラーを解決するには、このレベルを設定することをお勧めします。

このログ記録レベルでは、SDK は設定、認証情報解決、実行インターセプター、高レベルの TLS アクティビティ、リクエスト署名などに関する情報をログに記録します。

以下は、SDK が `S3Client#listBuckets()` 呼び出しに対して `DEBUG` レベルで出力するステートメントのサンプルです。

```
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 - (Sso0idcTokenProvider()) 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
...

```

次のlog4j2.xmlファイルは、前の出力を設定します。

```
<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>
```

## ワイヤログ記録を有効にする

SDK for Java 2.x によって送受信された詳細なリクエストおよびレスポンスの表示は役に立つ場合があります。この情報にアクセスする必要がある場合は、サービスクライアントが使用する HTTP クライアントに応じて必要な設定を追加することで、一時的に有効にできます。

デフォルトでは、[S3Client](#) などの同期サービスクライアントは基盤となる Apache を使用し [HttpClient](#)、[S3AsyncClient](#) などの非同期サービスクライアントは Netty ノンブロッキング HTTP クライアントを使用します。

2つのカテゴリーのサービスクライアントに使用できる HTTP クライアントの内訳は以下のとおりです。

同期 HTTP クライアント	非同期 HTTP クライアント
<a href="#">ApacheHttpClient</a> (デフォルト)	<a href="#">NettyNioAsyncHttpClient</a> (デフォルト)
<a href="#">URLConnectionHttpClient</a>	<a href="#">AwsCrtAsyncHttpClient</a>

基盤となる HTTP クライアントに応じて追加が必要な設定については、以下の該当するタブを参照してください。

**⚠ Warning**

ワイヤログファイルはデバッグの目的でのみ使用することをお勧めします。この機能を使用して機密データを記録することができるため、本稼働環境ではこの機能を無効にします。HTTPS 呼び出しの場合でも、リクエストやレスポンスは暗号化せずに記録されます。大きなリクエスト (例: にファイルをアップロードする Amazon S3) またはレスポンスの場合、詳細なワイヤログ記録はアプリケーションのパフォーマンスに大きな影響を与える可能性があります。

## ApacheHttpClient

「org.apache.http.wire」ロガーを log4j2.xml 設定ファイルに追加し、レベルを「DEBUG」に設定します。

次のlog4j2.xmlファイルは、Apache のフルワイヤログ記録をオンにします HttpClient。

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

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

Apache では内部で 1.2 を使用しているため、Apache でのワイヤログ記録には log4j-1.2-api アーティファクトへの Maven 依存関係がさらに必要です。

Apache HTTP クライアントのワイヤログ記録を含む log4j 2 の Maven 依存関係のすべてのセットは、以下の構築ファイルスニペットに示されています。

## 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")
}
...
```

2.20.0 アーティファクトの最小バージョンには log4j-bom を使用します。最新バージョンには、[Maven Central](#) に公開されているバージョンを使用します。VERSION は、使用するバージョンに置き換えます。

## URLConnectionHttpClient

URLConnectionHttpClient を使用するサービスクライアントの詳細をログ記録するには、まず以下の内容の logging.properties ファイルを作成します。

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

以下の JVM システムプロパティに logging.properties のフルパスを設定します。

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

この設定では、リクエストとレスポンスのヘッダーのみがログに記録されます。以下に例を示します。

```
<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=e367fa0bc217a6a65675bb743e1280cf12f8e8d566196a816d948fdf0b42ca1a}{User-
Agent: aws-sdk-java/2.17.230 Mac_OS_X/12.5 OpenJDK_64-Bit_Server_VM/25.332-b08
Java/1.8.0_332 vendor/Amazon.com_Inc. io/sync http/URLConnection cfg/retry-mode/
legacy}{x-amz-content-sha256: UNSIGNED-PAYLOAD}{X-Amz-Date: 20220927T133955Z}{x-amz-
te: append-md5}{Host: tkhill-test1.s3.amazonaws.com}{Accept: text/html, image/gif,
image/jpeg, *, q=.2, */*; q=.2}{Connection: keep-alive}
<Response> FINE: sun.net.www.MessageHeader@70a36a6611 pairs: {null: HTTP/1.1
200 OK}{x-amz-id-2: sAFeZD0KdUMsBbkDjyDZw7P0oocb4C9KbiuzfJ6TWKQsGXHM/
dFu0vr2tUb7Y1wEHGdJ3DSIxq0=}{x-amz-request-id: P9QW9SMZ97FKZ9X7}{Date: Tue,
27 Sep 2022 13:39:57 GMT}{Last-Modified: Tue, 13 Sep 2022 14:38:12 GMT}{ETag:
"2cbe5ad4a064cedec33b452bebf48032"}{x-amz-transfer-encoding: append-md5}{Accept-
Ranges: bytes}{Content-Type: text/plain}{Server: AmazonS3}{Content-Length: 67}
```

リクエスト/レスポンスの本文を表示するには、JVM プロパティに -Djavax.net.debug=all を追加します。この追加プロパティは、すべての SSL 情報を含む大量の情報をログ記録します。

ログコンソールまたはログファイル内で、"GET" または "POST" を検索すると、実際のリクエストとレスポンスを含むログのセクションにすばやく移動できます。リクエストに "Plaintext before ENCRYPTION" を、レスポンスに "Plaintext after DECRYPTION" を検索すると、ヘッダーと本文の全文が表示されます。

## NettyNioAsyncHttpClient

非同期サービスクライアントがデフォルト NettyNioAsyncHttpClient を使用している場合は、HTTP ヘッダーとリクエスト/レスポンスの本文をログ記録するロガーを log4j2.xml ファイルに 2 つ追加します。

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

以下はすべての log4j2.xml の例です。

```
<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>
```

これらの設定は、すべてのヘッダー詳細とリクエスト/レスポンス本文をログ記録します。

## AwsCrtAsyncHttpClient

AwsCrtAsyncHttpClient のインスタンスを使用するようにサービスクライアントを設定している場合は、JVM システムプロパティを設定するか、プログラムで詳細をログ記録できます。

Log to a file at "Debug" level

システムプロパティの使用:	プログラムによる:
---------------	-----------



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

```
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

システムプロパティの使用:

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

プログラムによる:

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

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

セキュリティ上の理由から、「Trace」レベルでは、`AwsCrtAsyncHttpClient` はレスポンスヘッダーのみをログ記録します。リクエストヘッダー、リクエストボディ、レスポンスボディはログ記録されません。

## DNS 名参照用の JVM TTL を設定する

Java 仮想マシン (JVM) は DNS 名参照をキャッシュします。JVM がホスト名を IP アドレスに解決すると、time-to-live(TTL) と呼ばれる指定された期間、IP アドレスがキャッシュされます。

AWS リソースは DNS 名エントリを使用するため、JVM の TTL 値を 5 秒に設定することをお勧めします。これにより、リソースの IP アドレスが変更されたときに、アプリケーションは DNS に対して再度クエリを実行することで、リソースの新しい IP アドレスを取得し、使用できるようになります。

一部の Java 設定では JVM のデフォルトの TTL が設定されるため、JVM が再起動されるまで、DNS エントリが更新されることはありません。したがって、アプリケーションの実行中に AWS リソースの IP アドレスが変更された場合、JVM を手動で再起動し、キャッシュされた IP 情報が更新されるまで、そのリソースを使用することはできません。この場合、キャッシュされた IP 情報が定期的に更新されるように JVM の TTL を設定することが極めて重要です。

## JVM TTL を設定する方法

JVM の TTL を変更するには、[networkaddress.cache.ttl](#) セキュリティプロパティ値を設定し、Java 8 の場合は \$JAVA\_HOME/jre/lib/security/java.security ファイルに、Java 11 以降の場合は \$JAVA\_HOME/conf/security/java.security ファイルに networkaddress.cache.ttl プロパティを設定します。

以下は、TTL キャッシュが 5 秒に設定された java.security ファイルからのスニペットです。

```
#
# 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
...
```

環境変数で表される JVM で実行されるすべてのアプリケーションは、\$JAVA\_HOME この設定を使用します。

## AWS SDK for Java 2.x のベストプラクティス

このセクションでは、SDK for Java 2.x を使用する際のベストプラクティスについて説明します。

### トピック

- [可能なかぎり SDK クライアントを再利用する](#)
- [クライアントオペレーションからの入カストリームを閉じます。](#)
- [パフォーマンステストに基づいて HTTP 構成を調整します。](#)
- [Netty ベースの HTTP クライアントに OpenSSL を使用](#)
- [API タイムアウトの設定](#)
- [メトリクスを使用](#)

## 可能なかぎり SDK クライアントを再利用する

各 SDK クライアントは独自の HTTP 接続プールを維持します。プールにすでに存在する接続を新しいリクエストで再利用できるため、新しい接続を確立する時間を短縮できます。効果的に使用されていない接続プールが多すぎることによるオーバーヘッドを避けるため、クライアントの 1 つのインスタンスを共有することをおすすめします。SDK クライアントはすべてスレッドセーフです。

クライアントインスタンスを共有したくない場合は、クライアントが必要ないときにインスタンスで `close()` を呼び出してリソースを解放してください。

### クライアントオペレーションからの入力ストリームを閉じます。

[S3Client#getObject](#) などのストリーミング操作では、[ResponseInputStream](#) を直接操作する場合は、次を実行することをお勧めします。

- できるだけ早く入力ストリームからすべてのデータを読み取ります。
- できるだけ早く入力ストリームを閉じます。

このような推奨を行うのは、入力ストリームは HTTP 接続からのデータの直接ストリームであり、ストリームからのデータがすべて読み取られてストリームが閉じられるまで、基盤となる HTTP 接続を再利用できないためです。これらのルールに従わないと、開いているが未使用の HTTP 接続を大量に割り当てて、クライアントがリソースを使い果たす可能性があります。

### パフォーマンステストに基づいて HTTP 構成を調整します。

SDK には、一般的なユースケースに適用される [デフォルト HTTP 設定](#) セットが用意されています。お客様には、ユースケースに基づいてアプリケーションの HTTP 構成を調整することをお勧めします。

手始めに、SDK には [スマート設定のデフォルト](#) 機能が用意されています。この機能は、バージョン 2.17.102 で始めることで使用できます。ユースケースに応じて、適切な設定値が得られるモードを選択します。

### Netty ベースの HTTP クライアントに OpenSSL を使用

デフォルトでは、SDK の [NettyNioAsyncHttpClient](#) は JDK のデフォルトの SSL 実装を `SslProvider` として使用します。テストの結果、OpenSSL は JDK のデフォルト実装よりもパフォーマンスが優れていることがわかりました。Netty コミュニティでは [OpenSSL の使用も推奨しています](#)。

OpenSSL を使用するには、依存関係に `netty-tcnative` を追加します。設定の詳細については、「[Netty プロジェクトのドキュメント](#)」を参照してください。

プロジェクト用に `netty-tcnative` を設定すると、`NettyNioAsyncHttpClient` インスタンスは自動的に OpenSSL を選択します。または、次のスニペットに示すように、`NettyNioAsyncHttpClient` ビルダーを使用して明示的に `SslProvider` を設定することもできます。

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

## API タイムアウトの設定

SDK は、接続タイムアウトやソケットタイムアウトなど、一部のタイムアウトオプションにはデフォルト値を提供しますが、API コールタイムアウトや個々の API コール試行タイムアウトには [デフォルト値](#) を提供しません。個々の試行とリクエスト全体の両方にタイムアウトを設定するのがグッドプラクティスです。これにより、リクエストの完了までの時間が長くなるような一時的な問題や、致命的なネットワーク上の問題が発生した場合に、アプリケーションが最適な方法で迅速に失敗するようになります。

[ClientOverrideConfiguration#apiCallAttemptTimeout](#) と [ClientOverrideConfiguration#apiCallTimeout](#) を使用して、サービスクライアントが行うすべてのリクエストにタイムアウトを設定できます。

次の例は、カスタムタイムアウト値を使用する Amazon S3 クライアントの設定を示しています。

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

### **apiCallAttemptTimeout**

この設定では、1 回の HTTP 試行にかかる時間を設定します。この時間が経過すると API コールを再試行できます。

### **apiCallTimeout**

このプロパティの値は、すべての再試行を含む実行全体の時間を設定します。

サービスクライアントでこれらのタイムアウト値を設定する代わりに、[RequestOverrideConfiguration#apiCallTimeout\(\)](#) と [RequestOverrideConfiguration#apiCallAttemptTimeout\(\)](#) を使用して 1 つのリクエストを設定できます。

次の例では、カスタムタイムアウト値を使用して 1 つの `listBuckets` リクエストを設定します。

```
s3Client.listBuckets(lbr -> lbr.overrideConfiguration(
    b -> b.apiCallTimeout(Duration.ofSeconds(<custom value>))
        .apiCallAttemptTimeout(Duration.ofMillis(<custom value>))));
```

これらのプロパティを一緒に使用すると、すべての再試行に費やされる合計時間にハードリミットを設定します。また、低速なリクエストに対して、フェイルファストするよう、個々の HTTP リクエストを設定することもできます。

## メトリクスを使用

SDK for Java は、アプリケーションのサービスクライアントの[メトリクスを収集](#)できます。これらのメトリクスを使用して、パフォーマンスの問題を特定したり、全体的な使用傾向を確認したり、返されたサービスクライアントの例外を確認したり、特定の問題を掘り下げて理解したりすることができます。

アプリケーションのパフォーマンスをより深く理解するために、メトリクスを収集して Amazon CloudWatch Logs を分析することをお勧めします。

## トラブルシューティングに関するよくある質問

AWS SDK for Java 2.x アプリケーションで を使用すると、このトピックに記載されているランタイムエラーが発生する可能性があります。根本原因を特定し、エラーを解決するには、こちらの提案を参考にしてください。

**java.net.SocketException 「: Connection reset」** または 「server failed to complete the response」 エラーを修正するにはどうすればよいですか？

接続リセットエラーは、リクエストが完了する前にホスト AWS のサービス、または中間パーティ (NAT ゲートウェイ、プロキシ、ロードバランサーなど) が接続を閉じたことを示します。ソリュー

シオンを見つけるには、接続が閉じている理由がわかっている必要があります。通常、次の項目が原因で接続が閉じられます。

- 接続は非アクティブです。これは、ストリーミング操作でよくあることです。この操作では、データが一定期間ワイヤとの間で書き込まれていないため、中間関係者は接続をデッドとして検出して閉じます。これを防ぐには、アプリケーションがデータをアクティブにダウンロードまたはアップロードしていることを確認してください。
- HTTP または SDK クライアントを閉じました。使用中はリソースを閉じないでください。
- プロキシの設定ミス。問題が解決したかどうかを確認するように設定したプロキシをバイパスしてみてください。この問題が修正されると、何らかの理由でプロキシが接続を閉じています。特定のプロキシを調べて、接続を閉じている理由を特定します。

問題を特定できない場合は、ネットワークのクライアントエッジで、影響を受ける接続の TCP ダンプを実行してみてください (例えば、制御するプロキシの後など)。

AWS エンドポイントが TCP RST (リセット) を送信していることがわかった場合は、[該当するサービスに連絡して](#)、リセットが行われている理由を判断できるかどうかを確認します。問題が発生したときのリクエスト IDs とタイムスタンプを提供する準備を整えます。また、AWS サポートチームは、アプリケーションが送受信しているバイト数とタイミングを正確に示す[ワイヤログ](#)からメリットを得ることもできます。

## 「接続タイムアウト」を修正するにはどうすればよいですか？

接続タイムアウトエラーは AWS のサービス、ホスト、または中間パーティ (NAT ゲートウェイ、プロキシ、ロードバランサーなど) が、設定された接続タイムアウト内にサーバーとの新しい接続を確立できなかったことを示します。この問題の一般的な原因を以下に示します。

- 設定された接続タイムアウトが低すぎます。デフォルトでは、の接続タイムアウトは 2 秒です AWS SDK for Java 2.x。接続タイムアウトの設定が低すぎると、このエラーが発生する可能性があります。推奨される接続タイムアウトは、リージョン内呼び出しのみを行う場合は 1 秒、クロスリージョンリクエストを行う場合は 3 秒です。
- プロキシの設定ミス。問題が解決したかどうかを確認するために設定したプロキシをバイパスしてみてください。この問題が解決した場合、プロキシが接続タイムアウトの理由になります。特定のプロキシを調べて、それが起こっている理由を調べる

問題を特定できない場合は、ネットワークのクライアントエッジで影響を受ける接続の TCP ダンプを (例えば、制御するプロキシの後などに) 実行して、ネットワークの問題を調査してください。

## java.net.SocketTimeoutException 「: 読み取りタイムアウト」を修正するにはどうすればよいですか？

読み取りタイムアウトエラーは、JVM が基盤となるオペレーティングシステムからデータを読み取ろうとしたが、データが SDK で設定された時間内に返されなかったことを示します。このエラーは AWS のサービス、オペレーティングシステム、または中間パーティ (NAT ゲートウェイ、プロキシ、ロードバランサーなど) が JVM が期待する時間内にデータを送信できない場合に発生する可能性があります。解決策を見つけるには、データが返されない原因が多数あるため、確認する必要があります。

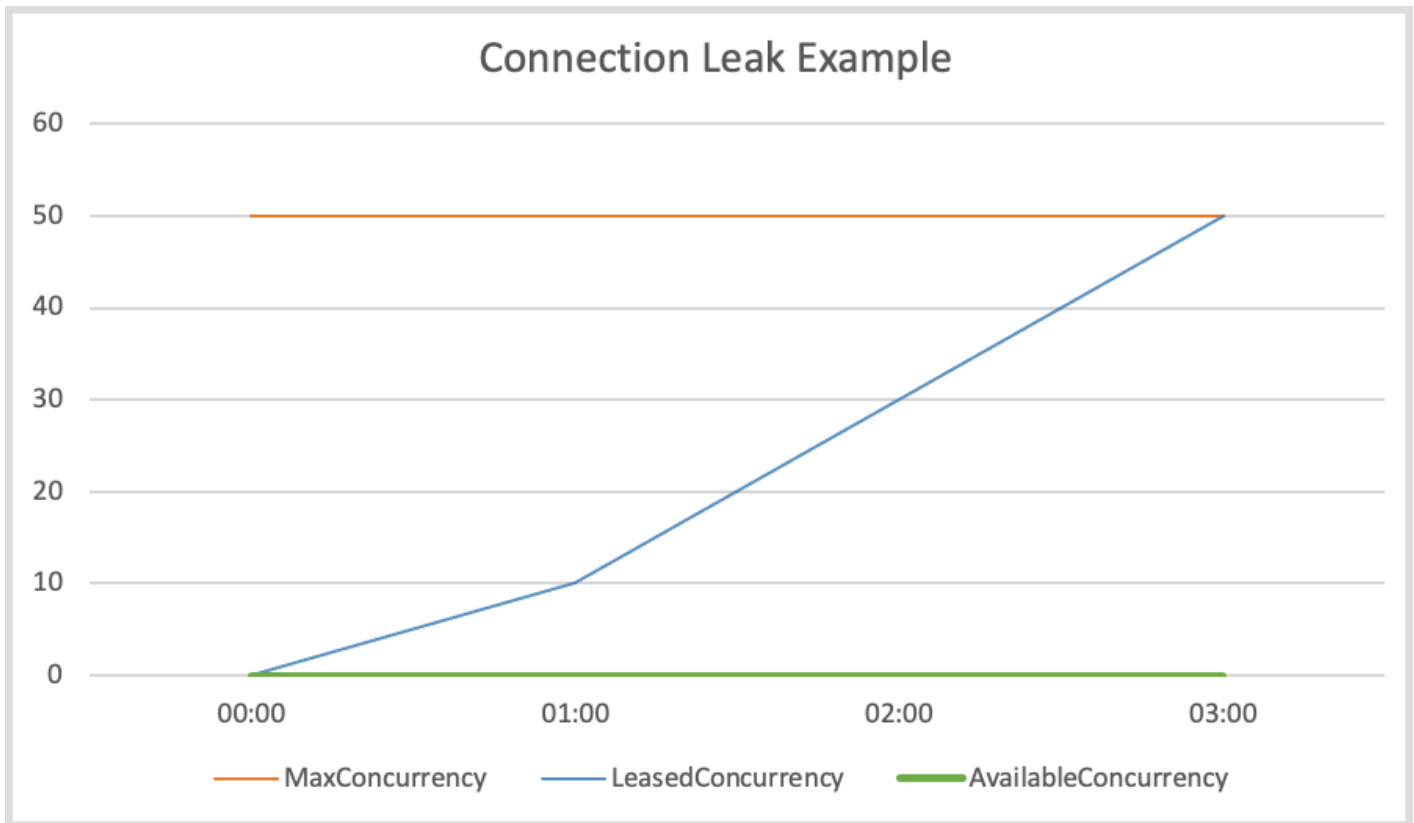
ネットワークのクライアントエッジで、影響を受ける接続の TCP ダンプを実行してみてください (例えば、制御するプロキシの後など)。

AWS エンドポイントが TCP RST (リセット) を送信していることがわかった場合は、[影響を受けるサービスにお問い合わせください](#)。問題が発生したときのリクエスト IDs とタイムスタンプを提供する準備を整えます。また、AWS サポートチームは、アプリケーションが送受信しているバイト数とタイミングを正確に示す[ワイヤログ](#)を利用することもできます。

## 「HTTP リクエストを実行できません: プールからの接続を待っているタイムアウト」エラーを修正するにはどうすればよいですか？

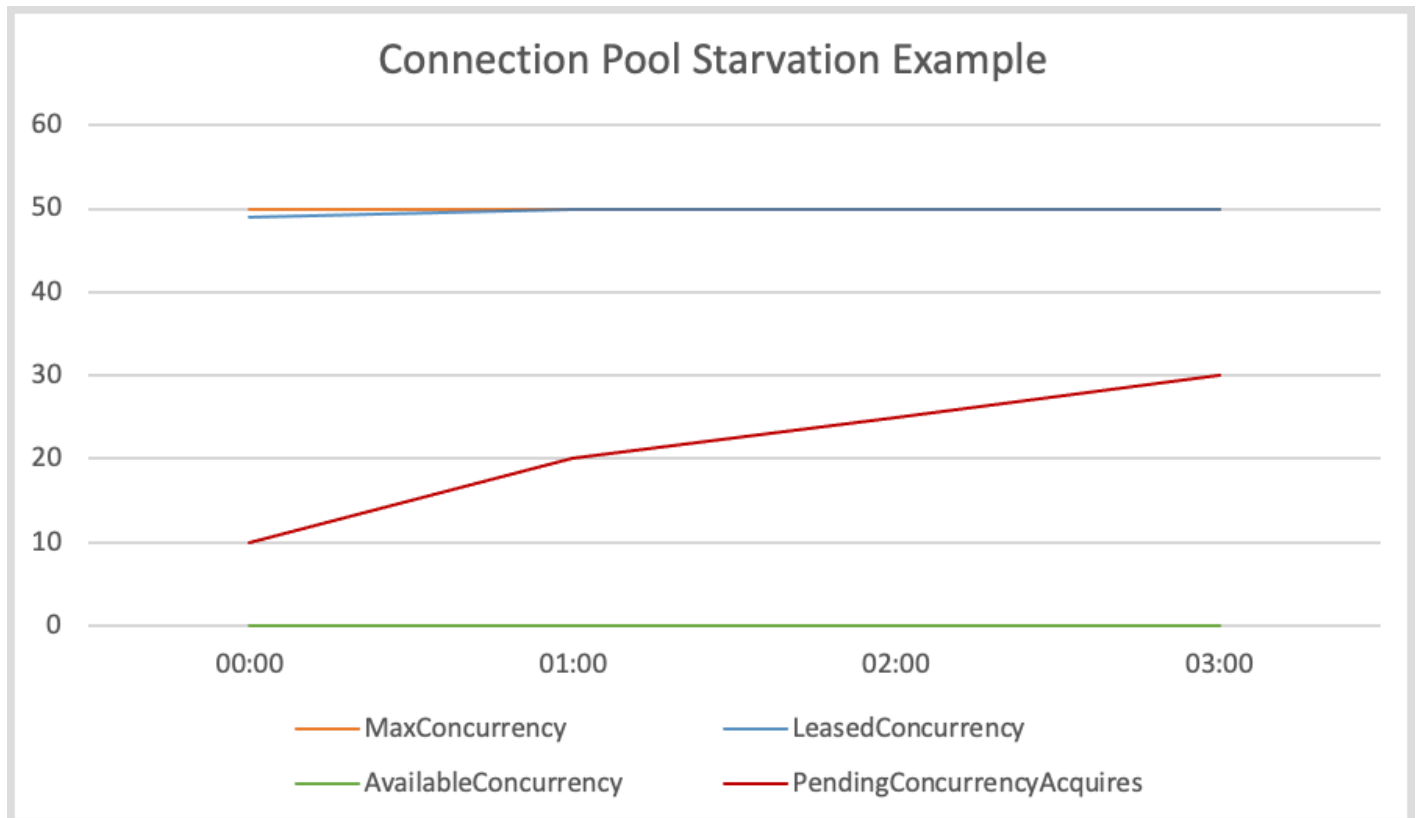
このエラーは、リクエストが指定された最大時間内にプールから接続を取得できないことを示します。問題のトラブルシューティングを行うには、[SDK クライアント側のメトリクスを有効にして Amazon にメトリクスを発行することをお勧めします](#) CloudWatch。HTTP メトリクスは、根本原因を絞り込むのに役立ちます。以下の項目では、このエラーの一般的な原因について説明します。

- 接続リーク。これを調査するには LeasedConcurrency、AvailableConcurrency、および MaxConcurrency メトリクスを確認します。に達するまで LeasedConcurrency が増加する MaxConcurrency が、減らすことがない場合、接続リークが発生する可能性があります。リークの一般的な原因は、S3 getObject メソッドなどのストリーミングオペレーションがクローズされていないことです。アプリケーションは、できるだけ早く入力ストリームからすべてのデータを読み取り、[その後で入力ストリームを閉じる](#) ことをお勧めします。次の表は、接続リークの SDK メトリクスがどのように表示されるかを示しています。



- 接続プールの枯渇。これは、リクエストレートが高く、設定されている接続プールのサイズがリクエストの需要を満たせない場合に発生する可能性があります。デフォルトの接続プールサイズは 50 で、プール内の接続が最大値に達すると、HTTP クライアントは接続が利用可能になるまで受信リクエストをキューに入れます。次の表は、接続プールの枯渇時に SDK メトリクスがどのように表示されるかを示しています。





この問題を軽減するには、次のいずれかのアクションを実行することを検討してください。

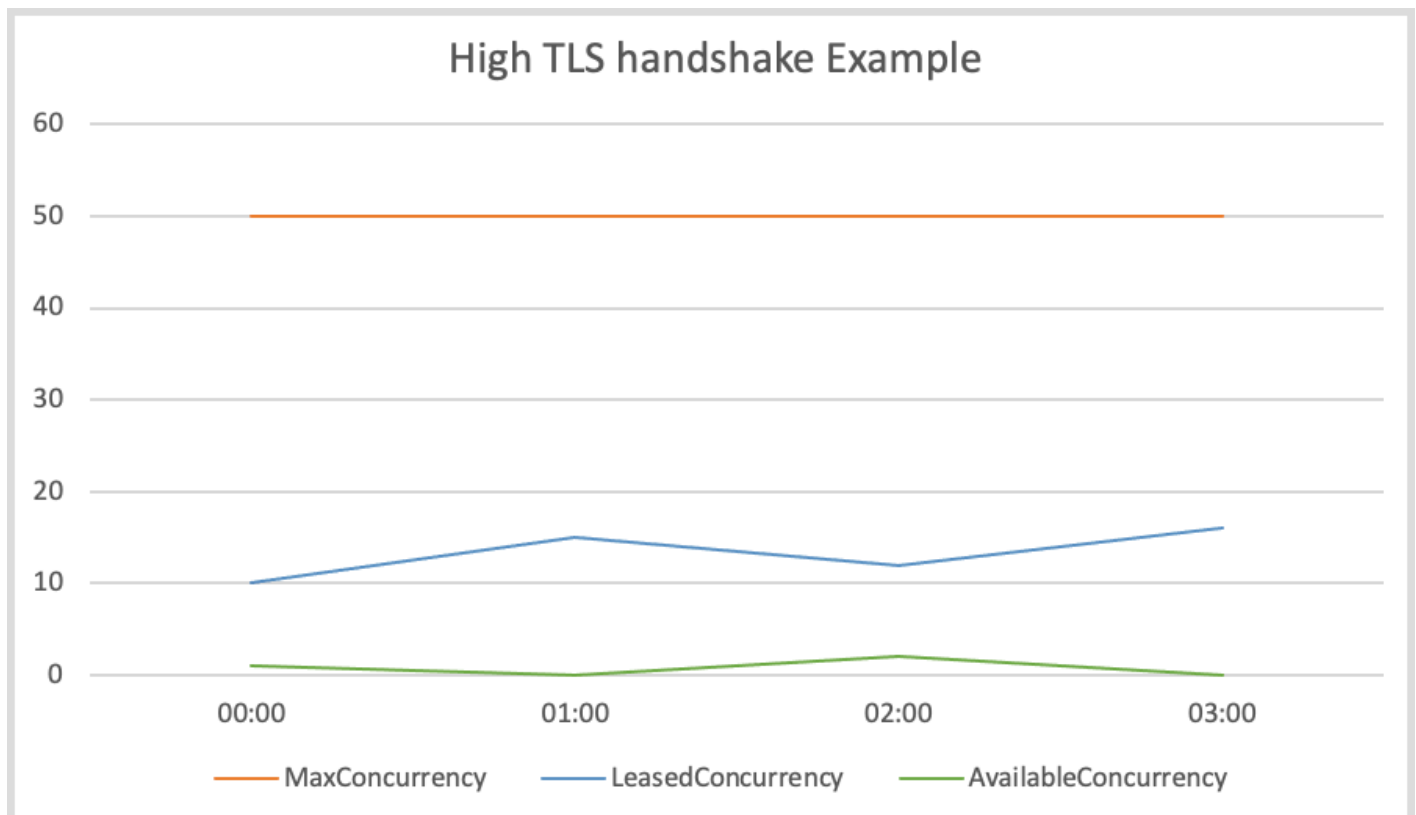
- 接続プールのサイズを増やします。
- 取得タイムアウトを増やします。
- リクエストレートを遅くします。

接続の最大数を増やすと、クライアントのスループットが向上します (ネットワークインターフェイスが既に完全に使用されていない限り)。ただし、最終的には、プロセスで使用されるファイル記述子の数に関するオペレーションシステムの制限に達する可能性があります。ネットワークインターフェイスを既に完全に使用しているか、接続数をさらに増やすことができない場合は、取得タイムアウトを増やしてみてください。この増加により、タイムアウトする前にリクエストが接続を取得するまでさらに時間がかかります。接続が解放されない場合、後続のリクエストは引き続きタイムアウトします。

最初の 2 つのメカニズムを使用して問題を解決できない場合は、次のオプションを試してリクエストレートを遅くします。

- 大量のトラフィックのバーストによってクライアントが過負荷にならないように、リクエストをスムーズ化します。
- への呼び出しをより効率的にします AWS のサービス。

- リクエストを送信するホストの数を増やします。
- I/O スレッドがビジー状態です。これは、`NettyNioAsyncHttpClient` を使用している場合にのみ適用されます。AvailableConcurrency メトリクスが低ではなく、接続がプールで使用可能であることを示しているが、高い場合は、I/O スレッドConcurrencyAcquireDurationがリクエストを処理できないことが原因である可能性があります。I/O スレッドがブロックされる可能性があるため、[将来の完了エグゼキューターRunnable:run](#)として渡さず、将来の応答の完了チェーンで時間のかかるタスクを実行しないようにしてください。そうでない場合は、`eventLoopGroupBuilder`メソッドを使用して I/O スレッドの数を増やすことを検討してください。参考までに、`NettyNioAsyncHttpClient` インスタンスの I/O スレッドのデフォルト数は、ホストの CPU コア数の 2 倍です。
- TLS ハンドシェイクのレイテンシーが高い。AvailableConcurrency メトリクスが 0 近く、よりもLeasedConcurrency低い場合はMaxConcurrency、TLS ハンドシェイクのレイテンシーが高いことが原因である可能性があります。次の表は、TLS ハンドシェイクレイテンシーが高い場合の SDK メトリクスの表示例を示しています。



CRT に基づいていない Java SDK が提供する HTTP クライアントの場合は、[TLS ログ](#)を有効にして TLS の問題をトラブルシューティングしてみてください。CRT AWS ベースの HTTP クライアントの場合は、[AWS CRT ログ](#) を有効にしてみてください。AWS エンドポイントが TLS ハンド

シェイクの実行に時間がかかると思われる場合は、[影響を受けるサービスに連絡](#)する必要があります。

## NoClassDefFoundError、NoSuchMethodErrorまたはを修正するにはどうすればよいですかNoSuchFieldError？

は、実行時にクラスをロードできなかつたNoClassDefFoundErrorことを示します。このエラーの最も一般的な原因は、次の2つです。

- JAR が見つからないか、JAR のバージョンがクラスパスにあるため、クラスがクラスパスに存在しません。
- 静的イニシャライザーが例外をスローしたため、クラスはロードに失敗しました。

同様に、と NoSuchFieldErrorは通常、JAR バージョンの不一致が原因NoSuchMethodErrorです。次のステップを実行することをお勧めします。

1. 依存関係をチェックして、すべての SDK jar の同じバージョンを使用していることを確認します。クラス、メソッド、またはフィールドが見つからない最も一般的な理由は、新しいクライアントバージョンにアップグレードしても、古い「共有」SDK 依存関係バージョンを引き続き使用する場合です。新しいクライアントバージョンは、新しい「共有」SDK 依存関係にのみ存在するクラスの使用を試みる場合があります。mvn dependency:tree または gradle dependencies (Gradle の場合) を実行して、SDK ライブラリのバージョンがすべて一致することを確認します。今後この問題を完全に回避するには、[BOM \(部品表\)](#) を使用して SDK モジュールバージョンを管理することをお勧めします。

次の例は、混合 SDK バージョンの例を示しています。

```
[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
```

のバージョンは 2.20.00 dynamodb で、のバージョンaws-coreは 2.13.19 です。aws-core アーティファクトのバージョンも 2.20.00 である必要があります。

2. ログの早い段階で ステートメントをチェックして、静的初期化の失敗が原因でクラスがロードに失敗したかどうかを確認します。クラスが初めてロードに失敗すると、クラスをロードできない理由を指定する別のより有用な例外がスローされる場合があります。この潜在的に有用な例外

は 1 回だけ発生するため、後続のログステートメントでは、クラスが見つからないことのみが報告されます。

3. デプロイプロセスをチェックして、必要な JAR ファイルがアプリケーションとともに実際にデプロイされていることを確認します。正しいバージョンで構築している可能性があります。アプリケーションのクラスパスを作成するプロセスでは、必要な依存関係が除外されます。

## SignatureDoesNotMatch 「」エラーまたは「計算したリクエスト署名が指定した署名と一致しません」エラーを修正するにはどうすればよいですか？

SignatureDoesNotMatch エラーは、によって生成された署名 AWS SDK for Java とによって生成された署名が一致し AWS のサービス ないことを示します。以下の項目では、考えられる原因について説明します。

- プロキシまたは仲介者がリクエストを変更します。例えば、プロキシまたはロードバランサーは、SDK によって署名されたヘッダー、パス、またはクエリ文字列を変更できます。
- サービスと SDK は、それぞれが署名する文字列を生成するときにリクエストをエンコードする方法が異なります。

この問題のデバッグを行うには、SDK の [デバッグログを有効にする](#) ことをお勧めします。エラーを再現して、SDK が生成した正規リクエストを見つけようとしています。ログでは、正規リクエストにはというラベルが付けられAWS4 Canonical Request: ...、署名する文字列には AWS4 String to sign: ... というラベルが付けられます。

デバッグを有効にすることができない場合、例えば、本番環境でのみ機能するため、エラーが発生したときにリクエストに関する情報をログに記録するロジックをアプリケーションに追加します。その後、その情報を使用して、デバッグログを有効にした統合テストでエラーを本番環境外にレプリケートできます。

正規リクエストと署名文字列を収集したら、[AWS 署名バージョン 4 仕様](#)と比較して、SDK が署名文字列を生成した方法に問題がないかどうかを判断します。問題が発生したと思われる場合は、[GitHub バグレポート](#)を作成できます AWS SDK for Java。

何も間違っていない場合は、署名する SDK の文字列と署名する文字列を比較して、失敗レスポンスの一部としてその文字列 (Amazon S3 など) AWS のサービス を返します。これが利用できない場合は、[影響を受ける サービスに連絡して](#)、それらが比較のために生成した正規リクエストと署名文字

列を確認する必要があります。これらの比較は、サービスとクライアント間のリクエストまたはエンコードの違いを変更した可能性のある中間パーティを特定するのに役立ちます。

リクエストの署名に関する背景情報については、「AWS Identity and Access Management ユーザーガイド」の [AWS「API リクエストの署名」](#) を参照してください。

#### Example 正規リクエストの

```
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 署名する文字列の

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

## java.lang.IllegalStateException 「: 接続プールのシャットダウン」エラーを修正するにはどうすればよいですか？

このエラーは、基盤となる Apache HTTP 接続プールが閉じられたことを示します。以下の項目では、考えられる原因について説明します。

- SDK クライアントは途中で閉じられました。SDK は、関連付けられたクライアントが閉じられた場合にのみ接続プールを閉じます。使用中はリソースを閉じないでください。
- **java.lang.Error** がスローされました。などのエラー `OutOfMemoryError` により、Apache HTTP 接続プールが [をシャットダウン](#) します。ログでエラースタックトレースを確認します。また、`Throwable` または `をキャッチするError` が、エラーが表示されない出力を隠している場所に

についてもコードを確認してください。コードでエラーが報告されない場合は、情報がログに記録されるようにコードを書き換えます。ログに記録された情報は、エラーの根本原因を特定するのに役立ちます。

- を閉じ `DefaultCredentialsProvider#create()` た後に から返された認証情報プロバイダーを使用しようとした。はシングルトンインスタンスを `DefaultCredentialsProvider#create` 返すため、閉じてコードが `resolveCredentials` メソッドを呼び出すと、キャッシュされた認証情報 (またはトークン) の有効期限が切れた後に例外がスローされます。

次の例に示すように、 が閉じ `DefaultCredentialsProvider` られている場所のコードを確認します。

- singleton インスタンスは、 を呼び出して閉じられます。  
`DefaultCredentialsProvider#close()`.

```
DefaultCredentialsProvider defaultCredentialsProvider =
    DefaultCredentialsProvider.create(); // Singleton instance returned.
AwsCredentials credentials = defaultCredentialsProvider.resolveCredentials();

// Make calls to AWS #####.

defaultCredentialsProvider.close(); // Explicit close.

// Make calls to AWS #####.

// After the credentials expire, either of the following calls eventually results
// in a "Connection pool shut down" exception.
credentials = defaultCredentialsProvider.resolveCredentials();
// Or
credentials = DefaultCredentialsProvider.create().resolveCredentials();
```

- ブロック `DefaultCredentialsProvider#create()` で を `try-with-resources` 呼び出します。

```
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();
```

コードがシングルトンインスタンスを閉

じ`DefaultCredentialsProvider.builder().build()`、を使用して認証情報を解決する必要がある場合は、を呼び出して新しい非シングルトンインスタンスを作成し  
ます`DefaultCredentialsProvider`。

# AWS SDK for Java 2.x の機能を使用する

## 一般的な機能

SDK for Java 2.x には、AWS のサービス に対するプログラミングを簡単にするいくつかの機能が含まれています。

- SDK では、[ページ分割された結果の取得](#)および[リソースのポーリング](#)の背後で複雑なメカニズムが使用されています。
- [ノンブロッキング I/O を使用した非同期プログラミング](#)は、パフォーマンスを向上させながら同時コードを記述するのに役立ちます。SDK には、可能な場合はレイテンシーの短縮などの [HTTP/2](#) の利点があります。
- Java SDK は、アプリケーションの動作状態を監視するのに役立つ[メトリクス](#)を生成できます。

## サービス固有の機能

前述の一般的な機能に加えて、Java SDK は特定の の機能を提供します AWS のサービス。

- Amazon S3 — SDK では、Amazon S3 での[ファイルやディレクトリの操作を簡素化するために](#)、S3 Transfer Manager が提供されています。SDK の標準非同期 S3 API の使用中に[パフォーマンスと信頼性を向上させる](#)ために、SDK は AWS CRT ベースの S3 クライアントを提供します。
- DynamoDB — [オブジェクト指向のマッピング機能](#)が DynamoDB 拡張クライアント API によって提供されます。[拡張ドキュメント API](#) を使用して、JSON スタイルのドキュメント指向データを使用します。
- IAM — IAM Policy Builder API では、[安全でオブジェクト指向の方法で IAM ポリシーを作成できます](#)。

## AWS SDK for Java 2.x を使用してページ分割された結果を処理する

多くの AWS オペレーションは、レスポンスオブジェクトが大きすぎて 1 回のレスポンスで返せない場合にページ分割された結果を返します。AWS SDK for Java 1.0 では、レスポンスには、結果の次のページを取得するために使用するトークンが含まれています。これとは対照的に、AWS SDK for Java 2.x には自動ページ分割メソッドがあり、複数のサービス呼び出しを実行して結果の次のページ



を自動的に取得します。この結果を処理するコードを記述するだけです。自動ページ分割は、同期クライアントと非同期クライアントの両方で使用できます。

### Note

これらのコードスニペットは、ユーザーが [SDK の使用の基本](#) を理解し、[シングルサインオンアクセス](#) で環境を設定していることを前提としています。

## 同期ページ割り

次の例は、Amazon S3 バケット内のオブジェクトを一覧表示する同期ページ分割メソッドを示しています。

### ページを反復処理する

最初の例は、ページネーターオブジェクト、[ListObjectsV2Iterable](#) インスタンスを使用して、`listRes stream` メソッドですべてのレスポンスページを反復処理する方法を示しています。コードはレスポンスページをストリーミングし、レスポンスストリームを [S3Object](#) コンテンツのストリームに変換して、Amazon S3 オブジェクトのコンテンツを処理します。

以下のインポートは、この同期ページ分割セクションのすべての例に適用されます。

### インポート

```
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()));
```

[の完全な例](#)を参照してください GitHub。

## オブジェクトを反復処理する

次の例では、レスポンスのページではなく、レスポンスで返るオブジェクトを反復処理する方法を示します。ListObjectsV2Iterable クラスの contents メソッドは、基になるコンテンツ要素を処理する複数のメソッドを提供する [SdkIterable](#) を返します。

### ストリーミングを使用する

以下のスニペットは、レスポンスコンテンツの stream メソッドを使用して、ページ分割された項目コレクションを反復処理します。

```
// Helper method to work with paginated collection of items directly.
listRes.contents().stream()
    .forEach(content -> System.out
```

```
        .println(" Key: " + content.key() + "
size = " + content.size()));
```

[の完全な例](#)を参照してください GitHub。

### for-each ループを使用する

SdkIterable は Iterable インターフェイスを拡張するので、他の Iterable と同じように内容を処理できます。次のスニペットでは、標準の for-each ループを使用してレスポンスの内容を反復処理しています。

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

[の完全な例](#)を参照してください GitHub。

### 手動のページ割り

手動のページ分割が必要なユースケースでは、従来どおり手動のページ分割を使用できます。それ以降のリクエストについては、応答オブジェクトの次のトークンを使用します。次の例では while ループを使用します。

```
        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();

}
```

[の完全な例](#)を参照してください GitHub。

## 非同期ページ割り

次の例は、DynamoDB テーブルを一覧表示する非同期ページ分割メソッドを示しています。

### テーブル名のページを反復処理する

次の 2 つの例では、`listTablesPaginator`メソッドを呼び出して取得する非同期 DynamoDB クライアントを使用しています [ListTablesPublisher](#)。は 2 つのインターフェイス `ListTablesPublisher` を実装しており、レスポンスを処理する多くのオプションを提供します。各インターフェイスのメソッドを見ていきます。

### Subscriber を使用する

次のコード例は、`ListTablesPublisher` で実装された `org.reactivestreams.Publisher` インターフェイスを使用してページ分割された結果を処理する方法を示しています。リアクティブストリームモデルの詳細については、[「リアクティブストリーム GitHub リポジトリ」](#)を参照してください。

以下のインポートは、この非同期ページ分割セクションのすべての例に適用されます。

### インポート

```
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;
```

次のコードは ListTablesPublisher インスタンスを取得します。

```
// 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);
```

次のコードでは、org.reactivestreams.Subscriber の匿名実装を使用して各ページの結果を処理します。

この onSubscribe メソッドでは、Subscription.request メソッドを呼び出して、公開者に対してデータのリクエストを開始します。公開者からデータの取得を開始するには、このメソッドを呼び出す必要があります。

サブスクライバーの onNext メソッドは、すべてのテーブル名にアクセスし、それぞれを出力することでレスポンスページを処理します。ページが処理されると、パブリッシャーに別のページが要求されます。このメソッドは、すべてのページが取得されるまで繰り返し呼び出されます。

データ取得中にエラーが発生すると、onError メソッドがトリガーされます。最後に、すべてのページがリクエストされると、onComplete メソッドが呼び出されます。

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

[の完全な例](#)を参照してください GitHub。

## Consumer を使用する

ListTablesPublisher を実装する SdkPublisher インターフェースには、Consumer を受け取って CompletableFuture<Void> を返す subscribe メソッドがあります。

このインターフェースの subscribe メソッドは、org.reactivestreams.Subscriber のオーバーヘッドが大きすぎるような単純なユースケースに使用できます。以下のコードは各ページを使用するため、それぞれの tableNames メソッドを呼び出します。tableNames メソッドは、forEach メソッドで処理される DynamoDB テーブル名の java.util.List を返します。

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

[の完全な例](#)を参照してください [GitHub](#)。

## テーブル名を反復処理する

次の例では、レスポンスのページではなく、レスポンスで返るオブジェクトを反復処理する方法を示します。前に `contents` メソッドで示した同期 Amazon S3 の例と同様、DynamoDB 非同期結果クラス、`ListTablesPublisher` には、基になるアイテムコレクションを操作するための `tableNames` 便利なメソッドがあります。`tableNames` メソッドの戻り型は、すべてのページに項目をリクエストできる [SdkPublisher](#) です。

### Subscriber を使用する

次のコードは、基になるテーブル名のコレクションの `SdkPublisher` を取得します。

```
// 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();
```

次のコードでは、`org.reactivestreams.Subscriber` の匿名実装を使用して各ページの結果を処理します。

サブスクライバーの `onNext` メソッドはコレクションの個々の要素を処理します。この場合はテーブル名です。テーブル名が処理されると、パブリッシャーから別のテーブル名が要求されます。このメソッドは、すべてのテーブル名が取得されるまで繰り返し呼び出されます。

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

[の完全な例](#)を参照してください GitHub。

## Consumer を使用する

次の例では、各項目の処理に Consumer を要する SdkPublisher の subscribe メソッドを使用しています。

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

[の完全な例](#)を参照してください GitHub。

## サードパーティーライブラリを使用する

カスタムの受信者を実装せずに、その他のサードパーティーのライブラリを使用することができます。この例では、[RxJava](#) の使用方法を示していますが RxJava、リアクティブストリームインターフェイスを実装するライブラリを使用することができます。そのライブラリの詳細については、[RxJava の Wiki ページ](#) [GitHub](#)を参照してください。

このライブラリを使用するには、依存関係として追加します。使用する POM スニペットの例を示します (Maven を使用している場合)。

## POM エントリ

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



```
</dependency>
```

## Code

```
DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();
ListTablesPublisher publisher =
asyncClient.listTablesPaginator(ListTablesRequest.builder()
    .build());

// The Flowable class has many helper methods that work with
// an implementation of an org.reactivestreams.Publisher.
List<String> tables = Flowable.fromPublisher(publisher)
    .flatMapIterable(ListTablesResponse::tableName)
    .toList()
    .blockingGet();
System.out.println(tables);
```

[の完全な例](#)を参照してください [GitHub](#)。

## AWS SDK for Java 2.x のリソース状態のポーリング: ウェーター

AWS SDK for Java 2.x のウェイターユーティリティを使用すると、AWS リソースに対してオペレーションを実行する前に、リソースが指定された状態であることを検証できます。

ウェーターは、DynamoDB テーブルや Amazon S3 バケットなどの AWS リソースが目的の状態に達するまで (またはリソースが目的の状態に到達しないと判断されるまで)、リソースをポーリングするために使用される抽象化です。面倒でエラーが発生しやすい AWS リソースを継続的にポーリングするロジックを記述する代わりに、ウェイターを使用してリソースをポーリングし、リソースの準備ができた後もコードを実行し続けることができます。

### 前提条件

でプロジェクトでウェーターを使用する前に AWS SDK for Java、[「AWS SDK for Java 2.x のセットアップ」](#) の手順を完了する必要があります。

AWS SDK for Javaのバージョン 2.15.0 以降を使用するように、プロジェクトの依存関係を (例: pom.xml または build.gradle ファイルで) 設定する必要があります。

例:

```
<project>
```

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

## ウェーターの使用

ウェイターオブジェクトをインスタンス化するには、最初にサービスクライアントを作成します。サービスクライアントの `waiter()` メソッドをウェイターオブジェクトの値として設定します。ウェイターインスタンスが存在するようになったら、適切なコードを実行するようにその応答オプションを設定します。

### 同期プログラミング

次のコードスニペットは、DynamoDB テーブルが存在し、ACTIVE 状態になるまで待機する方法を示しています。

```
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);
```

### 非同期プログラミング

次のコードスニペットは、DynamoDB テーブルが存在しなくなるのを待つ方法を示しています。

```
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();
```

## ウェイターを設定する

ビルダーで `overrideConfiguration()` を使用して、ウェイターの設定をカスタマイズできます。一部の操作では、リクエストを実行する際にカスタム設定を適用できます。

### ウェイターを設定する

次のコードスニペットは、ウェイターの設定を上書きする方法を示しています。

```
// 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();
```

### 特定のリクエストの設定をオーバーライドする

次のコードスニペットは、リクエストごとにウェイターの設定を上書きする方法を示しています。すべての操作でカスタマイズ可能な設定があるわけではないことに注意してください。

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

asyncWaiter.waitUntilTableExists(b -> b.tableName("myTable"),
```

```
o -> o.waitTimeout(Duration.ofMinutes(1));
```

## コードの例

でウェーターを使用する完全な例については DynamoDB、AWS 「コード例リポジトリ」の「[.CreateTablejava](#)」を参照してください。

でウェーターを使用する完全な例については Amazon S3、AWS 「コード例リポジトリ」の[S3BucketOps.java](#)」を参照してください。

## 非同期プログラミングを使用する

は、数スレッドに高い同時実行性を実装するノンブロッキング I/O サポートを備えた非同期クライアント AWS SDK for Java 2.x を備えています。ただし、ノンブロッキング I/O の合計は保証されません。非同期クライアントは、認証情報の取得、[AWS 署名バージョン 4 \(SigV4\)](#) を使用したリクエスト署名、エンドポイント検出など、場合によってはブロック呼び出しを実行することがあります。

同期メソッドは、クライアントがサービスからのレスポンスを受信するまでスレッドの実行をブロックします。非同期メソッドはすぐに応答を返し、レスポンスを待機せずに呼び出しスレッドに制御を戻します。

非同期メソッドはレスポンスが可能になる前に応答を返すため、準備ができたらレスポンスを得るための手段が必要になります。2.x の AWS SDK for Java 戻り `CompletableFuture` オブジェクトの非同期クライアントのメソッド。準備ができたらレスポンスにアクセスできます。

## 非ストリーミング操作

非ストリーミングオペレーションの場合、非同期メソッド呼び出しは同期メソッドに似ています。ただし、の非同期メソッドは、将来の の非同期オペレーションの結果を含む [CompletableFuture](#) オブジェクト AWS SDK for Java を返します。

結果が出力される場合に完了するアクションを指定して、`CompletableFuture whenComplete()` メソッドを呼び出します。`CompletableFuture` には `Future` インターフェイスが実装されているため、あわせて `get()` メソッドを呼び出すことでレスポンスオブジェクトも取得できます。

以下は、Amazon DynamoDB 関数を呼び出してテーブルのリストを取得し、[ListTablesResponse](#) オブジェクトを保持 `CompletableFuture` できる を受け取る非同期オペレーションの例です。非同期の呼び出しが完了した場合にのみ、`whenComplete()` への呼び出しで定義されるアクションが行われます。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;
import java.util.concurrent.CompletableFuture;
```

## Code

```
public class DynamoDBAsyncListTables {

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

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

        listTables(client);
    }

    public static void listTables(DynamoDbAsyncClient client) {

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

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

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

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

次のコード例は、非同期クライアントを使用してテーブルから項目を取得する方法を示しています。の `getItem` メソッドを呼び出し `DynamoDbAsyncClient`、目的の項目のテーブル名とプライマリキー値を含む [GetItemRequest](#) オブジェクトを渡します。これは通常、オペレーションに必要なデータを渡す方法です。この例では、文字列値が渡されることに注意してください。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

## Code

```
public static void getItem(DynamoDbAsyncClient client, String tableName, String
key, String keyVal) {

    HashMap<String, AttributeValue> keyToGet =
        new HashMap<String, AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal).build());

    try {
```

```
// Create a GetItemRequest instance
GetItemRequest request = GetItemRequest.builder()
    .key(keyToGet)
    .tableName(tableName)
    .build();

// Invoke the DynamoDbAsyncClient object's getItem
java.util.Collection<AttributeValue> returnedItem =
client.getItem(request).join().item().values();

// Convert Set to Map
Map<String, AttributeValue> map =
returnedItem.stream().collect(Collectors.toMap(AttributeValue::s, s->s));
Set<String> keys = map.keySet();
for (String sinKey : keys) {
    System.out.format("%s: %s\n", sinKey, map.get(sinKey).toString());
}

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

[の完全な例](#)を参照してください [GitHub](#)。

## ストリーミング操作

ストリーミングオペレーションでは、コンテンツを段階的に提供[AsyncRequestBody](#)するための、またはレスポンスを受信して処理[AsyncResponseTransformer](#)するための を指定する必要があります。

次の例では、PutObjectオペレーションを使用してファイルを Amazon S3 非同期的に にアップロードします。

### インポート

```
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import java.nio.file.Paths;
import java.util.concurrent.CompletableFuture;
```

## Code

```
/**
 * To run this AWS code example, ensure that you have setup your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class S3AsyncOps {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "Usage:\n" +
            "  S3AsyncOps <bucketName> <key> <path>\n\n" +
            "Where:\n" +
            "  bucketName - the name of the Amazon S3 bucket (for example,
bucket1). \n\n" +
            "  key - the name of the object (for example, book.pdf). \n" +
            "  path - the local path to the file (for example, C:/AWS/book.pdf).
\n" ;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String key = args[1];
        String path = args[2];

        Region region = Region.US_WEST_2;
        S3AsyncClient client = S3AsyncClient.builder()
            .region(region)
            .build();

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();
```



```
// Put the object into the bucket
CompletableFuture<PutObjectResponse> future = client.putObject(objectRequest,
    AsyncRequestBody.fromFile(Paths.get(path))
);
future.whenComplete((resp, err) -> {
    try {
        if (resp != null) {
            System.out.println("Object uploaded. Details: " + resp);
        } else {
            // Handle error
            err.printStackTrace();
        }
    } finally {
        // Only close the client when you are completely done with it
        client.close();
    }
});

future.join();
}
```

次の例では、GetObjectオペレーションを使用して Amazon S3 から非同期的にファイルを取得します。

## インポート

```
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.nio.file.Paths;
import java.util.concurrent.CompletableFuture;
```

## Code

```
/**
 * To run this AWS code example, ensure that you have setup your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class S3AsyncStreamOps {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "Usage:\n" +
            "  S3AsyncStreamOps <bucketName> <objectKey> <path>\n\n" +
            "Where:\n" +
            "  bucketName - the name of the Amazon S3 bucket (for example,
bucket1). \n\n" +
            "  objectKey - the name of the object (for example, book.pdf). \n" +
            "  path - the local path to the file (for example, C:/AWS/book.pdf).
\n" ;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String path = args[2];

        Region region = Region.US_WEST_2;
        S3AsyncClient client = S3AsyncClient.builder()
            .region(region)
            .build();

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        CompletableFuture<GetObjectResponse> futureGet =
client.getObject(objectRequest,
    AsyncResponseTransformer.toFile(Paths.get(path)));

        futureGet.whenComplete((resp, err) -> {
            try {
                if (resp != null) {
                    System.out.println("Object downloaded. Details: "+resp);
                }
            }
        });
    }
}
```

```
        } else {
            err.printStackTrace();
        }
    } finally {
        // Only close the client when you are completely done with it
        client.close();
    }
});
futureGet.join();
}
}
```

## 高度な操作

AWS SDK for Java 2.x は、非同期イベント駆動型ネットワークアプリケーションフレームワークである [Netty](#) を使用して I/O スレッドを処理します。AWS SDK for Java 2.x は Netty の背後に `ExecutorService` を作成し、HTTP クライアントリクエストから Netty クライアントに返された `Future` を完了します。この抽象化により、開発者がスレッドを停止またはスリープすることを選択した場合、アプリケーションが非同期プロセスをブレイクするリスクが軽減されます。デフォルトでは、各非同期クライアントはプロセッサ数に基づいてスレッドプールを作成し、`ExecutorService` 内のキュー内のタスクを管理します。

上級ユーザーは、構築時に次のオプションを使用して、非同期クライアントを作成するときにスレッドプールサイズを指定できます。

### Code

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

パフォーマンスを最適化するには、独自のスレッドプールエグゼキュターを管理し、クライアントの設定時に含めます。

```
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();
```

## で HTTP/2 を使用する AWS SDK for Java

HTTP/2 は HTTP プロトコルのメジャーバージョンです。この新しいバージョンには、パフォーマンスを向上させるために次のようないくつかの機能強化が図られています。

- より効率的なデータ転送を実現するバイナリデータのエンコーディング。
- ヘッダーの圧縮により、クライアントによってダウンロードされるオーバーヘッドバイトが減り、クライアントにコンテンツをより速く提供できます。これは特に、既に帯域幅の制限を受けているモバイルクライアントで役立ちます。
- 双方向非同期通信 (多重化) により、クライアントと間の複数のリクエストと応答メッセージが、複数の接続ではなく 1 つの接続で同時に処理 AWS されるため、パフォーマンスが向上します。

最新の SDK にアップグレードする開発者は、使用中のサービスでサポートされている場合、自動的に HTTP/2 を使用します。新しいプログラミングインターフェイスはシームレスに HTTP/2 を活用して、アプリケーションを構築するための新しい方法を提供します。

AWS SDK for Java 2.x には、HTTP/2 プロトコルを実装するイベントストリーミング用の新しい APIs が含まれています。これらの新しい API の使用方法の例については、[Kinesis の使用](#)を参照してください。

## から SDK メトリクスを使用する AWS SDK for Java

AWS SDK for Java 2.x を使用すると、アプリケーション内のサービスクライアントに関するメトリクスを収集し、で出力を分析し Amazon CloudWatch、それに基づいてアクションを実行できます。

デフォルトでは、SDK でメトリクスの収集が無効になっています。このトピックは、それを有効にして設定するのに役立ちます。

### 前提条件

メトリクスを有効にして使用する前に、次の手順を完了する必要があります。

- 「[セットアップ](#)」のステップを完了します。
- AWS SDK for Javaのバージョン 2.14.0 以降を使用するように、プロジェクトの依存関係を (例: pom.xml または build.gradle ファイルで) 設定します。

へのメトリクスの発行を有効にするには CloudWatch、プロジェクトの依存関係にcloudwatch-metric-publisherバージョン番号 2.14.0 以降の artifactId も含めます。

例:

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

- メトリクスパブリッシャーが使用する IAM ID の `cloudwatch:PutMetricData` アクセス許可を有効にして、SDK for Java がメトリクスを書き込めるようにします。

## メトリクスの収集を有効にする方法

アプリケーションでサービスクライアントのメトリクスを有効にすることも、個々のリクエストでメトリクスを有効にすることもできます。

### 特定のリクエストのメトリクスを有効にする

次のクラスは、へのリクエストに対して CloudWatch メトリクスパブリッシャーを有効にする方法を示しています Amazon DynamoDB。デフォルトのメトリクスパブリッシャー設定を使用します。

```
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

サービスクライアントが使用されなくなったときに、アプリケーションが [MetricPublisher](#) インスタンス `close` を呼び出すことを確認します。そうしないと、スレッドまたはファイル記述子のリークが発生する可能性があります。

## 特定のサービスクライアントの概要メトリクスを有効にする

次のコードスニペットは、サービスクライアントのデフォルト設定で CloudWatch メトリクスパブリッシャーを有効にする方法を示しています。

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

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

## メトリクスパブリッシャーをカスタマイズする

次のクラスは、特定のサービスクライアントのメトリクスパブリッシャーのカスタム設定を設定する方法を示しています。カスタマイズには、特定のプロファイルのロード、メトリクスパブリッシャーがリクエストを送信する AWS リージョンの指定、パブリッシャーがメトリクスを送信する頻度のカスタマイズが含まれます CloudWatch。

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

import java.time.Duration;

public class CustomConfigForDDBClient {
    // Use one MetricPublisher for your application. It can be used with requests or
    service clients.
```

```
static MetricPublisher metricsPub = CloudWatchMetricPublisher.builder()
    .cloudWatchClient(CloudWatchAsyncClient.builder()
        .region(Region.US_WEST_2)
        .credentialsProvider(ProfileCredentialsProvider.create("cloudwatch"))
        .build())
    .uploadFrequency(Duration.ofMinutes(5))
    .maximumCallsPerUpload(100)
    .namespace("ExampleSDKV2Metrics")
    .detailedMetrics(CoreMetric.API_CALL_DURATION)
    .build();

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

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

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

前のスニペットで示したカスタマイズには、次の効果があります。

- `cloudWatchClient` メソッドを使用すると、メトリクスの送信に使用する CloudWatch クライアントをカスタマイズできます。この例では、クライアントがメトリクスを送信するデフォルトの `us-east-1` とは異なるリージョンを使用します。また、別の名前付きプロファイルである `cloudwatch` も使用します。このプロファイルの認証情報は、へのリクエストの認証に使用されます CloudWatch。これらの認証情報には、へのアクセス許可が必要です `cloudwatch:PutMetricData`。



- `uploadFrequency` メソッドを使用すると、メトリクスパブリッシャーがメトリクスを にアップロードする頻度を指定できます CloudWatch。デフォルトは 1 分に 1 回です。
- `maximumCallsPerUpload` メソッドは、アップロードごとに行われる呼び出しの数を制限します。デフォルトは無制限です。
- デフォルトでは、SDK for Java 2.x は名前空間 の下にメトリクスを発行します `AwsSdk/JavaSdk2`。 `namespace` メソッドを使用して、別の値を指定できます。
- デフォルトでは、SDK はサマリーメトリクスを発行します。概要メトリクスは、平均、最小、最大、合計、サンプル数で構成されます。 `detailedMetrics` メソッドで 1 つ以上の SDK メトリクスを指定することで、SDK はメトリクスごとに追加のデータを発行します。この追加データにより、でクエリできる p90 や p99 などのパーセンタイル統計が有効になります CloudWatch。詳細なメトリクスは `APICallDuration`、SDK クライアントリクエストのレイテンシーを測定するなどの end-to-end レイテンシーメトリクスに特に役立ちます。 [CoreMetric](#) クラスのフィールドを使用して、他の一般的な SDK メトリクスを指定できます。

## メトリクスが利用可能になるまでの時間

メトリクスは通常、SDK for Java によって発行されてから 5 ~ 10 分以内に利用可能になります。正確な と up-to-date メトリクスについては、Java アプリケーションからメトリクスを送信してから少なくとも 10 分後に Cloudwatch を確認してください。

## 収集される情報

メトリクスの収集には、次のものが含まれます。

- API リクエストの数 (成功したか、失敗したかを含む)
- 返される例外など、API リクエストで呼び出す AWS サービスに関する情報
- Marshalling、Signing、HTTP リクエストなどのさまざまな操作の期間
- 開いている接続の数、保留中のリクエストの数、使用されている HTTP クライアントの名前などの HTTP クライアントのメトリクス

### Note

使用可能なメトリクスは、HTTP クライアントによって異なります。

詳細なリストについては、[サービスクライアントのメトリクス](#)を参照してください。

## この情報の使用方法

SDK が収集するメトリクスを使用して、アプリケーションのサービスクライアントをモニタリングできます。全体的な使用傾向の確認や、異常の特定ができるほか、返されたサービスクライアントの例外を確認したり、特定の問題を理解するために詳しく確認したりすることもできます。を使用して Amazon CloudWatch、アプリケーションが定義した条件に達したらすぐに通知するアラームを作成することもできます。

詳細については、「[Amazon CloudWatch ユーザーガイド](#)」の [Amazon CloudWatch 「メトリクスの使用」](#) と [Amazon CloudWatch 「アラームの使用」](#) を参照してください。

## サービスクライアントのメトリクス

を使用すると AWS SDK for Java 2.x、アプリケーションのサービスクライアントからメトリクスを収集し、それらのメトリクスを [Amazon に発行 \(出力\) CloudWatch](#) できます。

これらのテーブルは、収集できるメトリクスと HTTP クライアントの使用要件を示しています。

SDK のメトリクスの有効化と設定の詳細については、[SDK メトリクスの有効化](#) を参照してください。

## 各リクエストで収集されたメトリクス

メトリクス名	説明	[Type] (タイプ)
ApiCallDuration	リクエストの完了にかかる合計時間 (すべての再試行を含む)。	duration
ApiCallSuccessful	API コールが成功した場合は true、そうでない場合は false。	ブール値
CredentialsFetchDuration	リクエスト AWS の署名認証情報を取得するのにかかる時間。	duration
EndpointResolveDuration	API コールに使用されるエンドポイントの解決にかかった時間。	duration

メトリクス名	説明	[Type] (タイプ)
MarshallingDuration	SDK リクエストを HTTP リクエストにマーシャリングするのにかかる時間。	duration
OperationName	リクエストが行われた AWS API の名前。	文字列
RetryCount	SDK が API コールを再試行した回数。	整数
ServiceId	API リクエスト AWS のサービスが行われたのサービス ID。	文字列
TokenFetchDuration	リクエストのトークン署名認証情報を取得するのにかかる時間。	duration

## リクエストの各試行について収集されたメトリクス

各 API コールでは、レスポンスが受信されるまでに複数回試行する必要がある場合があります。これらのメトリクスは、各試行について収集されます。

### コアメトリクス

メトリクス名	説明	[Type] (タイプ)
AwsExtendedRequestId	サービスリクエストの拡張リクエスト ID。	文字列
AwsRequestId	サービスリクエストのリクエスト ID。	文字列
BackoffDelayDuration	この API コールが試行されるまでに SDK が待機した時間。	duration

メトリクス名	説明	[Type] (タイプ)
ErrorType	通話の試行で発生したエラーのタイプ。	文字列
ReadThroughput	クライアントの読み取りスループット。	ダブル
ServiceCallDuration	サービスへの接続、リクエストの送信、レスポンスからの HTTP ステータスコードとヘッダーの受信にかかる時間。	duration
SigningDuration	HTTP リクエストの署名にかかる時間。	duration
TimeToFirstByte	HTTP リクエストの送信 (接続の取得を含む) からレスポンス内のヘッダーの最初のバイトの受信までの経過時間。	duration
TimeToLastByte	HTTP リクエストの送信 (接続の取得を含む) からレスポンスの最後のバイトの受信までの経過時間。	duration
UnmarshallingDuration	SDK レスポンスへの HTTP レスポンスのマーシャリングを解除するのにかかる時間。	duration

## HTTP メトリクス

メトリクス名	説明	[Type] (タイプ)	HTTP クライアントが必要*
AvailableConcurrency	別の接続を確立することなく、HTTP クラ	整数	アパッチ、Netty、CRT

メトリクス名	説明	[Type] (タイプ)	HTTP クライアントが必要*
	クライアントでサポートできる残りの同時リクエストの数。		
ConcurrencyAcquireDuration	接続プールからチャネルを取得するのにかかる時間。	duration	アパッチ、Netty、CRT
HttpClientName	リクエストに使用されている HTTP の名前。	文字列	アパッチ、Netty、CRT
HttpStatusCode	HTTP レスポンスとともに返されるステータスコード。	整数	すべて
LeasedConcurrency	HTTP クライアントによって現在実行されているリクエストの数。	整数	アパッチ、Netty、CRT
LocalStreamWindowSize	このリクエストが実行されたストリームのローカル HTTP/2 ウィンドウサイズ。	整数	Netty
MaxConcurrency	HTTP クライアントでサポートされる同時リクエストの最大数。	整数	アパッチ、Netty、CRT

メトリクス名	説明	[Type] (タイプ)	HTTP クライアントが必要*
PendingConcurrencyAcquires	別の TCP 接続または新しいストリームが接続プールから利用可能になるまで待機して、ブロックされたリクエストの数。	整数	アパッチ、Netty、CRT
RemoteStreamWindowSize	このリクエストが実行されたストリームのリモート HTTP/2 ウィンドウサイズをバイト単位で指定します。	整数	Netty

列で使用される用語の意味は次のとおりです。

- Apache: アパッチベースの HTTP クライアント ([ApacheHttpClient](#))
- Netty: Netty ベースの HTTP クライアント ([NettyNioAsyncHttpClient](#))
- CRT: AWS CRT ベースの HTTP クライアント ([AwsCrtAsyncHttpClient](#))
- すべて: メトリクスデータの収集は HTTP クライアントに依存しません。これには [URLConnection](#) ベースの HTTP クライアント ([URLConnectionHttpClient](#)) が含まれます。

# AWS のサービスを使用した の操作 AWS SDK for Java 2.x

このセクションでは、選択した の操作方法に関する短いチュートリアルとガイダンスを提供します AWS のサービス。完全な例については、[「コード例」セクション](#)を参照してください。

## トピック

- [CloudWatch を使用する](#)
- [AWS データベースサービスと AWS SDK for Java 2.x の操作 DynamoDB](#)
- [の操作 Amazon EC2](#)
- [IAM を使用する](#)
- [の操作 Kinesis](#)
- [AWS Lambda 関数の呼び出し、一覧表示、および削除](#)
- [Amazon S3 で使用する](#)
- [Amazon Simple Notification Service を使用する](#)
- [の操作 Amazon Simple Queue Service](#)
- [Amazon Transcribe を使用する](#)

## CloudWatch を使用する

このセクションでは、AWS SDK for Java 2.x を使用して [Amazon CloudWatch](#) をプログラムする例を示します。

Amazon CloudWatch は、Amazon Web Services (AWS) リソースと、AWS でリアルタイムに実行されるアプリケーションを監視します。リソースとアプリケーションで測定する変数であるメトリクスを収集し、追跡するには、CloudWatch を使用できます。CloudWatch アラームは、定義するルールに従って通知を送信し、また、モニタリングしているリソースに自動的に変更を行います。

次の例には各手法を示すのに必要なコードのみが含まれます。[完全なサンプルコードは GitHub で入手できます](#)。そこから、単一のソースファイルをダウンロードするカリポジトリをローカルにクローン作成して、ビルドし実行するためのすべての例を取得できます。

## トピック

- [CloudWatch からメトリクスを取得](#)

- [カスタムメトリクスデータを CloudWatch にパブリッシュ](#)
- [CloudWatch アラームを操作する](#)
- [Amazon CloudWatch Events を使用する](#)

## CloudWatch からメトリクスを取得

### メトリクスの一覧表示

CloudWatch メトリクスを一覧表示するには、を作成し[ListMetricsRequest](#)、CloudWatchClientのlistMetricsメソッドを呼び出します。ListMetricsRequest を使用して、名前空間、メトリクス名、またはディメンションで返されたメトリクスをフィルタリングできます。

#### Note

AWS サービスによって投稿されたメトリクスとディメンションの一覧は、「Amazon CloudWatch ユーザーガイド」の「[Amazon CloudWatch Metrics and Dimensions Reference](#)」で参照できます。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
```

### Code

```
public static void listMets( CloudWatchClient cw, String namespace) {

    boolean done = false;
    String nextToken = null;

    try {
        while(!done) {
```



```
ListMetricsResponse response;

if (nextToken == null) {
    ListMetricsRequest request = ListMetricsRequest.builder()
        .namespace(namespace)
        .build();

    response = cw.listMetrics(request);
} else {
    ListMetricsRequest request = ListMetricsRequest.builder()
        .namespace(namespace)
        .nextToken(nextToken)
        .build();

    response = cw.listMetrics(request);
}

for (Metric metric : response.metrics()) {
    System.out.printf(
        "Retrieved metric %s", metric.metricName());
    System.out.println();
}

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

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

メトリクスは、`getMetrics`メソッドを呼び出し [ListMetricsResponse](#) で返されます。

結果はページ分割される場合があります。結果の次のバッチを取得するには、レスポンスオブジェクトの `nextToken` を呼び出し、そのトークン値を使用して、新しいリクエストオブジェクトをビルドします。次に、`listMetrics` メソッドを新しいリクエストで再度呼び出します。

「」で [完全な例](#) を参照してください [GitHub](#)。

## 詳細情報

- [ListMetrics](#) Amazon CloudWatch API リファレンスの

## カスタムメトリクスデータを CloudWatch にパブリッシュ

複数の AWS のサービスが「AWS」で始まる名前空間で[独自のメトリクス](#)を発行します。（「AWS」で始まらない限り）独自の名前空間を使用してカスタムメトリクスデータを発行することもできます。

### カスタムメトリクスデータのパブリッシュ

独自のメトリクスデータを発行するには、[メトリクスデータ](#)を使用して CloudWatchClient の putMetricData メソッドを呼び出します [PutMetricDataRequest](#)。には、データに使用するカスタム名前空間と、[MetricDatum](#) オブジェクト内のデータポイント自体に関する情報を含める PutMetricDataRequest 必要があります。

#### Note

「AWS」で始まる名前空間を指定することはできません。「AWS」で始まる名前空間は、Amazon Web Services 製品による利用のために予約されています。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
```

## Code

```
public static void putMetData(CloudWatchClient cw, Double dataPoint ) {
```

```
try {
    Dimension dimension = Dimension.builder()
        .name("UNIQUE_PAGES")
        .value("URLS")
        .build();

    // Set an Instant object
    String time =
        ZonedDateTime.now( ZoneOffset.UTC ).format( DateTimeFormatter.ISO_INSTANT );
    Instant instant = Instant.parse(time);

    MetricDatum datum = MetricDatum.builder()
        .metricName("PAGES_VISITED")
        .unit(StandardUnit.NONE)
        .value(dataPoint)
        .timestamp(instant)
        .dimensions(dimension).build();

    PutMetricDataRequest request = PutMetricDataRequest.builder()
        .namespace("SITE/TRAFFIC")
        .metricData(datum).build();

    cw.putMetricData(request);

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.printf("Successfully put data point %f", dataPoint);
}
```

「」で[完全な例](#)を参照してください GitHub。

## 詳細情報

- ユーザーガイドの[Amazon CloudWatch 「メトリクスを使用するAmazon CloudWatch」](#)。
- Amazon CloudWatch ユーザーガイドの [AWS 名前空間](#)。
- [PutMetricData](#) Amazon CloudWatch API リファレンスの。

# CloudWatch アラームを操作する

## アラームの作成

CloudWatch メトリクスに基づいてアラームを作成するには、アラーム条件が [PutMetricAlarmRequest](#) 満たされた で CloudWatchClient の putMetricAlarm メソッドを呼び出します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
```

### Code

```
public static void putMetricAlarm(CloudWatchClient cw, String alarmName, String
instanceId) {

    try {
        Dimension dimension = Dimension.builder()
            .name("InstanceId")
            .value(instanceId).build();

        PutMetricAlarmRequest request = PutMetricAlarmRequest.builder()
            .alarmName(alarmName)
            .comparisonOperator(
                ComparisonOperator.GREATER_THAN_THRESHOLD)
            .evaluationPeriods(1)
            .metricName("CPUUtilization")
            .namespace("AWS/EC2")
            .period(60)
            .statistic(Statistic.AVERAGE)
            .threshold(70.0)
            .actionsEnabled(false)
            .alarmDescription(
                "Alarm when server CPU utilization exceeds 70%")
            .unit(StandardUnit.SECONDS)
```

```
        .dimensions(dimension)
        .build();

        cw.putMetricAlarm(request);
        System.out.printf(
            "Successfully created alarm with name %s", alarmName);

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

「」で[完全な例](#)を参照してください GitHub。

## アラームの一覧表示

作成したCloudWatchアラームを一覧表示するには、結果のオプションを設定するために[DescribeAlarmsRequest](#)使用できる で CloudWatchClientの describeAlarmsメソッドを呼び出します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
```

### Code

```
public static void desCWAAlarms( CloudWatchClient cw) {

    try {

        boolean done = false;
        String newToken = null;

        while(!done) {
            DescribeAlarmsResponse response;

            if (newToken == null) {
```

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

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

    if(response.nextToken() == null) {
        done = true;
    } else {
        newToken = response.nextToken();
    }
}

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

アラームのリストは、によって返[DescribeAlarmsResponse](#)される `MetricAlarms`で を呼び出すことで取得できます[describeAlarms](#)。

結果はページ分割される場合があります。結果の次のバッチを取得するには、レスポンスオブジェクトの `nextToken` を呼び出し、そのトークン値を使用して、新しいリクエストオブジェクトをビルドします。次に、`describeAlarms` メソッドを新しいリクエストで再度呼び出します。

#### Note

`CloudWatchClient`の `describeAlarmsForMetric`メソッドを使用して、特定のメトリクスのアラームを取得することもできます。使用方法は `describeAlarms` と同様です。

「」で[完全な例](#)を参照してください [GitHub](#)。

## アラームの削除

CloudWatch アラームを削除するには、削除するアラームの名前を 1 つ以上 [DeleteAlarmsRequest](#) 含む `deleteAlarms` の `CloudWatchClient` メソッドを呼び出します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
```

### Code

```
public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.printf("Successfully deleted alarm %s", alarmName);
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

「」で [完全な例](#) を参照してください GitHub。

### 詳細情報

- Amazon CloudWatch ユーザーガイドの [Amazon CloudWatch アラームの使用](#)
- [PutMetricAlarm](#) Amazon CloudWatch API リファレンスの
- [DescribeAlarms](#) Amazon CloudWatch API リファレンスの
- [DeleteAlarms](#) Amazon CloudWatch API リファレンスの

## Amazon CloudWatch Events を使用する

CloudWatch イベントは、AWS リソースの変更を示すシステムイベントのほぼリアルタイムのストリームを、Amazon EC2 インスタンス、Lambda 関数、Kinesis ストリーム、Amazon ECS タスク、Step Functions ステートマシン、Amazon SNS トピック、Amazon SQS キュー、または組み込みターゲットに配信します。簡単なルールを使用して、一致したイベントを 1 つ以上のターゲット関数またはストリームに振り分けることができます。

Amazon EventBridge は CloudWatch イベントの[進化](#)です。どちらのサービスも同じ API を使用するため、SDK が提供する[CloudWatch イベントクライアント](#)を引き続き使用することも、SDK for Java の CloudWatch イベント機能用[EventBridge クライアント](#)に移行することもできます。CloudWatch Events [ユーザーガイドのドキュメント](#)と [API リファレンス](#)は、EventBridge ドキュメントサイトから利用できるようになりました。

### イベントの追加

カスタム CloudWatch イベントを追加するには、各イベントの詳細を提供する 1 つ以上の[PutEventsRequest](#)オブジェクトを含む [PutEventsRequestEntry](#) オブジェクトで `CloudWatchEventsClient` の `sputEvents` メソッドを呼び出します。イベントのソースとタイプ、イベントに関連付けられたリソースなど、エントリの複数のパラメータを指定できます。

#### Note

`putEvents` への呼び出しごとに最大 10 個のイベントを指定できます。

### インポート

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;
```

### Code

```
public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn ) {

    try {

        final String EVENT_DETAILS =
```



```
        "{ \"key1\": \"value1\", \"key2\": \"value2\" }";

        PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
            .detail(EVENT_DETAILS)
            .detailType("sampleSubmitted")
            .resources(resourceArn)
            .source("aws-sdk-java-cloudwatch-example")
            .build();

        PutEventsRequest request = PutEventsRequest.builder()
            .entries(requestEntry)
            .build();

        cwe.putEvents(request);
        System.out.println("Successfully put CloudWatch event");

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

「」で[完全な例](#)を参照してください GitHub。

## ルールの追加

ルールを作成または更新するには、ルールの名前と[イベントパターン](#)、[ルールに関連付けるロール](#)、ルールの実行頻度を説明する[スケジュール式](#)などのオプションパラメータ `PutRuleRequest` を使用して、`CloudWatchEventsClient` の `sputRule` メソッドを呼び出します。IAM

## インポート

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;
```

## Code

```
public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
roleArn) {
```

```
try {
    PutRuleRequest request = PutRuleRequest.builder()
        .name(ruleName)
        .roleArn(roleArn)
        .scheduleExpression("rate(5 minutes)")
        .state(RuleState.ENABLED)
        .build();

    PutRuleResponse response = cwe.putRule(request);
    System.out.printf(
        "Successfully created CloudWatch events rule %s with arn %s",
        roleArn, response.ruleArn());
} catch (
    CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

「」で[完全な例](#)を参照してください GitHub。

## ターゲットの追加

ターゲットは、ルールがトリガーされたときに呼び出されるリソースです。ターゲットの例には、Amazon EC2 インスタンス、Lambda 関数、Kinesis ストリーム、Amazon ECS タスク、Step Functions ステートマシン、組み込みターゲットなどがあります。

ルールにターゲットを追加するには、更新するルール [PutTargetsRequest](#) を含むと、ルールに追加するターゲットのリストを使用して `CloudWatchEventsClient` の `sputTargets` メソッドを呼び出します。

## インポート

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.Target;
```

## Code

```
public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName, String
functionArn, String targetId ) {
```

```
try {
    Target target = Target.builder()
        .arn(functionArn)
        .id(targetId)
        .build();

    PutTargetsRequest request = PutTargetsRequest.builder()
        .targets(target)
        .rule(ruleName)
        .build();

    PutTargetsResponse response = cwe.putTargets(request);
    System.out.printf(
        "Successfully created CloudWatch events target for rule %s",
        ruleName);
} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

「」で[完全な例](#)を参照してください GitHub。

## 詳細情報

- [「Amazon ユーザーガイド」の「でのイベントの追加 PutEvents EventBridge」](#)
- [Amazon EventBridge ユーザーガイドのルールのスケジュール式](#)
- [「Amazon ユーザーガイド」の「のイベントタイプ CloudWatch Events EventBridge」](#)
- [Amazon EventBridge ユーザーガイドのイベントパターン](#)
- [PutEvents](#) 「Amazon EventBridge API リファレンス」の「」
- [PutTargets](#) 「Amazon EventBridge API リファレンス」の「」
- [PutRule](#) 「Amazon EventBridge API リファレンス」の「」

## AWS データベースサービスと AWS SDK for Java 2.x

AWS には、リレーショナル、キーバリュー、インメモリ、ドキュメントなど、[いくつかのデータベースタイプがあります](#)。SDK for Java 2.x のサポートは、のデータベースサービスの性質によって異なります AWS。

[Amazon DynamoDB](#) サービスなどの一部のデータベースサービスには、AWS リソース (データベース) を管理するウェブサービス APIs と、データを操作するウェブサービス APIs。Java 2.x 用 SDK では、これらのタイプのサービスには [DynamoDBClient](#) などの専用のサービスクライアントがあります。

他のデータベースサービスには、[Amazon DocumentDB](#) API (クラスター、インスタンス、リソース管理用) など、リソースとやり取りするウェブサービス API がありますが、データを操作するためのウェブサービス API はありません。SDK for Java 2.x には、リソースを操作するための対応する [DocDbClient](#) インターフェイスがあります。ただし、データを処理するには、[MongoDB for Java](#) などの別の Java API が必要です。

以下の例を使用して、さまざまなタイプのデータベースで Java 2.x サービスクライアント用 SDK を使用方法を説明します。

## Amazon DynamoDB の例

データの使用

SDK サービスクライアント: [DynamoDbClient](#)

例: [DynamoDB を使用した React/Spring REST アプリケーション](#)

例: [いくつかの DynamoDB の例](#)

SDK サービスクライアント: [DynamoDbEnhancedClient](#)

例: [DynamoDB を使用した React/Spring REST アプリケーション](#)

例: [いくつかの DynamoDB の例](#) (「拡張」で始まる名前)

データベースの使用

SDK サービスクライアント: [DynamoDbClient](#)

例: [CreateTable、ListTables、DeleteTable](#)

このガイドのガイド付きコード例のセクションにある「[additional DynamoDB examples](#)」を参照してください。

## Amazon RDS での例

データの使用	データベースの使用
非 SDK API: JDBC、データベース固有の SQL フレーバー。コードはデータベース接続または接続プールを管理します。	SDK サービスクライアント: <a href="#">RdsClient</a>
例: <a href="#">MySQL を使った React/Spring REST アプリケーション</a>	例: <a href="#">いくつかの RdsClient 例</a>

## Amazon Redshiftの例

データの使用	データベースの使用
SDK サービスクライアント: <a href="#">RedshiftDataClient</a>	SDK サービスクライアント: <a href="#">RedshiftClient</a>
例: <a href="#">いくつかの RedshiftDataClient 例</a>	例: <a href="#">いくつかの RedshiftClient 例</a>
例: <a href="#">を使用した React/Spring REST アプリケーション RedshiftDataClient</a>	

## Amazon Aurora Serverless v2 の例

データの使用	データベースの使用
SDK サービスクライアント: <a href="#">RdsDataClient</a>	SDK サービスクライアント: <a href="#">RdsClient</a>
例: <a href="#">を使用した React/Spring REST アプリケーション RdsDataClient</a>	例: <a href="#">いくつかの RdsClient 例</a>

## Amazon DocumentDB の例

データの使用	データベースの使用
非 SDK API: MongoDB 固有の Java ライブラリ ( <a href="#">MongoDB for Java</a> など)。コードはデータベース接続または接続プールを管理します。	SDK サービスクライアント: <a href="#">DocDbClient</a>
例: <a href="#">DocumentDB (Mongo) デベロッパーガイド</a> (「Java」タブを選択)	

## の操作 DynamoDB

このセクションでは、[DynamoDB](#) の操作方法の例を示します。

次の例では、AWS SDK for Java 2.x の標準の低レベル DynamoDB クライアント ([DynamoDbClient](#)) を使用します。

- [the section called “でのテーブルの操作 DynamoDB”](#)
- [the section called “DynamoDB の項目の操作”](#)

SDK には、DynamoDB を操作するための高レベルのオブジェクト指向アプローチを提供する [DynamoDB 拡張クライアント](#) も用意されています。次のセクションでは、このクライアントについて詳しく説明します。

- [the section called “DynamoDB アイテムにオブジェクトをマッピングする”](#)

## でのテーブルの操作 DynamoDB

テーブルは、DynamoDB データベース内のすべての項目のコンテナです。からデータを追加または削除する前に DynamoDB、テーブルを作成する必要があります。

テーブルごとに、以下を定義する必要があります。

- アカウントとリージョンに固有のテーブル名。
- プライマリキー。すべての値は一意でなければならず、テーブル内のどの 2 つの項目も同じプライマリキー値を持つことはできません。

プライマリキーは、単一のパーティション (ハッシュ) キーで構成されるシンプルなプライマリキーにするか、パーティションとソート (範囲) キーで構成される複合プライマリキーにすることができます。

各キー値には、[ScalarAttributeType](#) クラスによって列挙されたデータ型 が関連付けられています。キー値は、バイナリ (B)、数値 (N)、または文字列 (S) になります。詳細については、「Amazon DynamoDB デベロッパーガイド」の「[命名規則とデータ型](#)」を参照してください。

- テーブル用に予約された読み込み/書き込みキャパシティーユニットの数を定義するプロビジョンドスループットの値。

#### Note

[Amazon DynamoDB 料金](#)はテーブルに設定したプロビジョニングされたスループット値に基づいているため、テーブルに必要な容量だけを予約してください。

テーブルのプロビジョンドスループットはいつでも変更できるため、必要に応じてキャパシティーを調整できます。

## テーブルを作成する

DynamoDbClient's `createTable` メソッドを使用して新しい DynamoDB テーブルを作成します。テーブルのプライマリキーを識別するために使用する、テーブル属性とテーブルスキーマを構築する必要があります。また、最初のプロビジョニングされたスループット値およびテーブル名を指定する必要があります。

#### Note

選択した名前のテーブルが既に存在する場合は、[DynamoDbException](#) がスローされます。

## シンプルプライマリキーを使用してテーブルを作成する

このコードは、テーブルのシンプルなプライマリキーである属性を 1 つ持つテーブルを作成します。この例では、に [AttributeDefinition](#) および [KeySchemaElement](#) オブジェクトを使用しています [CreateTableRequest](#)。

## インポート

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

## Code

```
public static String createTable(DynamoDbClient ddb, String tableName, String key)
{
    DynamoDbWaiter dbWaiter = ddb.waiter();
    CreateTableRequest request = CreateTableRequest.builder()
        .attributeDefinitions(AttributeDefinition.builder()
            .attributeName(key)
            .attributeType(ScalarAttributeType.S)
            .build())
        .keySchema(KeySchemaElement.builder()
            .attributeName(key)
            .keyType(KeyType.HASH)
            .build())
        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10))
            .build())
        .tableName(tableName)
        .build();

    String newTable = "";
    try {
        CreateTableResponse response = ddb.createTable(request);
    }
```



```
DescribeTableRequest tableRequest = DescribeTableRequest.builder()
    .tableName(tableName)
    .build();

// Wait until the Amazon DynamoDB table is created
WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
waiterResponse.matched().response().ifPresent(System.out::println);

newTable = response.tableDescription().tableName();
return newTable;

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
```

「」で[完全な例](#)を参照してください GitHub。

## 複合プライマリキーを使用してテーブルを作成する

次の例では、2つの属性を持つテーブルを作成します。どちらの属性も複合プライマリキーに使用されます。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
```

### Code

```
public static String createTableComKey(DynamoDbClient ddb, String tableName) {
```

```
        CreateTableRequest request = CreateTableRequest.builder()
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("Language")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("Greeting")
                    .attributeType(ScalarAttributeType.S)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("Language")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("Greeting")
                    .keyType(KeyType.RANGE)
                    .build())
            .provisionedThroughput(
                ProvisionedThroughput.builder()
                    .readCapacityUnits(new Long(10))
                    .writeCapacityUnits(new Long(10)).build())
            .tableName(tableName)
            .build();

        String tableId = "";

        try {
            CreateTableResponse result = ddb.createTable(request);
            tableId = result.tableDescription().tableId();
            return tableId;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

「」で[完全な例](#)を参照してください [GitHub](#)。

## テーブルの一覧表示

特定のリージョンのテーブルを一覧表示するには、`DynamoDbClient` の `listTables` メソッドを呼び出します。

### Note

アカウントとリージョンの名前付きテーブルが存在しない場合は、[ResourceNotFoundException](#) がスローされます。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.List;
```

## Code

```
public static void listAllTables(DynamoDbClient ddb){

    boolean moreTables = true;
    String lastName = null;

    while(moreTables) {
        try {
            ListTablesResponse response = null;
            if (lastName == null) {
                ListTablesRequest request = ListTablesRequest.builder().build();
                response = ddb.listTables(request);
            } else {
                ListTablesRequest request = ListTablesRequest.builder()
                    .exclusiveStartTableName(lastName).build();
                response = ddb.listTables(request);
            }

            List<String> tableNames = response.tableNames();

            if (tableNames.size() > 0) {
```

```
        for (String curName : tableNames) {
            System.out.format("* %s\n", curName);
        }
    } else {
        System.out.println("No tables found!");
        System.exit(0);
    }

    lastName = response.lastEvaluatedTableName();
    if (lastName == null) {
        moreTables = false;
    }
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
System.out.println("\nDone!");
}
```

デフォルトでは、呼び出しごとに最大 100 個のテーブルが返されます。評価された最後のテーブルを取得するには、返された [ListTablesResponse](#) オブジェクト `lastEvaluatedTableName` を使用します。この値を使用して、前回の一覧表示で返された最後の値以降から、一覧表示を開始できます。

「」で [完全な例](#) を参照してください [GitHub](#)。

## テーブルの説明 (テーブルに関する情報の取得)

`DynamoDbClient` の `describeTable` メソッドを使用して、テーブルに関する情報を取得します。

### Note

アカウントとリージョンの名前付きテーブルが存在しない場合は、[ResourceNotFoundException](#) がスローされます。

## インポート

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

```
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;
```

## Code

```
public static void describeDynamoDBTable(DynamoDbClient ddb, String tableName ) {

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

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

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

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

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

            for (AttributeDefinition a : attributes) {
```

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

「」で[完全な例](#)を参照してください GitHub。

## テーブルの変更 (更新)

テーブルのプロビジョンドスループット値は、DynamoDbClient's updateTable メソッドを呼び出すことで随時変更できます。

### Note

アカウントとリージョンの名前付きテーブルが存在しない場合は、[ResourceNotFoundException](#) がスローされます。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.UpdateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

## Code

```
public static void updateDynamoDBTable(DynamoDbClient ddb,
                                       String tableName,
                                       Long readCapacity,
                                       Long writeCapacity) {

    System.out.format(
        "Updating %s with new provisioned throughput values\n",
        tableName);
}
```

```
System.out.format("Read capacity : %d\n", readCapacity);
System.out.format("Write capacity : %d\n", writeCapacity);

ProvisionedThroughput tableThroughput = ProvisionedThroughput.builder()
    .readCapacityUnits(readCapacity)
    .writeCapacityUnits(writeCapacity)
    .build();

UpdateTableRequest request = UpdateTableRequest.builder()
    .provisionedThroughput(tableThroughput)
    .tableName(tableName)
    .build();

try {
    ddb.updateTable(request);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

System.out.println("Done!");
}
```

「」で[完全な例](#)を参照してください GitHub。

## テーブルを削除する

テーブルを削除するには、DynamoDbClient's deleteTable メソッドを呼び出してテーブルの名前を指定します。

### Note

アカウントとリージョンの名前付きテーブルが存在しない場合は、[ResourceNotFoundException](#) がスローされます。

## インポート

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

## Code

```
public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

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

    try {
        ddb.deleteTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}
```

「」で[完全な例](#)を参照してください GitHub。

## 詳細情報

- Amazon DynamoDB デベロッパーガイドの[テーブルの操作のガイドライン](#)
- Amazon DynamoDB デベロッパーガイドの「[でのテーブルの操作 DynamoDB](#)」

## DynamoDB の項目の操作

DynamoDB で、項目とは、属性のコレクションで、それぞれに名前と値があります。属性値はスカラー型、セット型、ドキュメント型のいずれかです。詳細については、Amazon DynamoDB デベロッパーガイドの[命名規則とデータ型](#)を参照してください。

### テーブルからの項目の取り出し (取得)

DynamoDbClientの `getItem` メソッドを呼び出し、目的の項目のテーブル名とプライマリキーバリューを含む [GetItemRequest](#) オブジェクトを渡します。その項目のすべての属性を含む [GetItemResponse](#) オブジェクトを返します。特定の属性を取得するには、[で、1 つ以上の](#)プロジェクト式 `GetItemRequest` を指定します。

返された `GetItemResponse` オブジェクトの `item()` メソッドを使用して、項目に関連付けられているキー (文字列) と値 ([AttributeValue](#)) のペアの [マップ](#) を取得できます。



## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
```

## Code

```
public static void getDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal ) {

    HashMap<String, AttributeValue> keyToGet = new HashMap<String, AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal).build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName(tableName)
        .build();

    try {
        Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();

        if (returnedItem != null) {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");

            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
            }
        } else {
            System.out.format("No item found with the key %s!\n", key);
        }
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }  
}
```

「」で[完全な例](#)を参照してください GitHub。

## 非同期クライアントを使用したテーブルからの項目の取り出し (取得)

の `getItem` メソッドを呼び出し `DynamoDbAsyncClient`、目的の項目のテーブル名とプライマリキーバリューを含む [GetItemRequest](#) オブジェクトを渡します。

その項目のすべての属性を持つ [Collection](#) インスタンスを返すことができます (次の例を参照)。

### インポート

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;  
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;  
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Set;  
import java.util.stream.Collectors;  
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

### Code

```
public static void getItem(DynamoDbAsyncClient client, String tableName, String  
key, String keyVal) {  
  
    HashMap<String, AttributeValue> keyToGet =  
        new HashMap<String, AttributeValue>();  
  
    keyToGet.put(key, AttributeValue.builder()  
        .s(keyVal).build());  
  
    try {  
  
        // Create a GetItemRequest instance  
        GetItemRequest request = GetItemRequest.builder()  
            .key(keyToGet)  
            .tableName(tableName)  
            .build();
```

```
// Invoke the DynamoDbAsyncClient object's getItem
java.util.Collection<AttributeValue> returnedItem =
client.getItem(request).join().item().values();

// Convert Set to Map
Map<String, AttributeValue> map =
returnedItem.stream().collect(Collectors.toMap(AttributeValue::s, s->s));
Set<String> keys = map.keySet();
for (String sinKey : keys) {
    System.out.format("%s: %s\n", sinKey, map.get(sinKey).toString());
}

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

「」で[完全な例](#)を参照してください [GitHub](#)。

## テーブルへの新しい項目の追加

項目の属性を表すキーと値のペアの[マップ](#)を作成します。これらには、テーブルのプライマリキーフィールドの値を含める必要があります。プライマリキーで特定される項目がすでにある場合、フィールドはリクエストによって更新されます。

### Note

アカウントとリージョンの名前付きテーブルが存在しない場合は、[ResourceNotFoundException](#) がスローされます。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;
```

## Code

```
public static void putItemInTable(DynamoDbClient ddb,
                                String tableName,
                                String key,
                                String keyVal,
                                String albumTitle,
                                String albumTitleValue,
                                String awards,
                                String awardVal,
                                String songTitle,
                                String songTitleVal){

    HashMap<String,AttributeValue> itemValues = new
HashMap<String,AttributeValue>();

    // Add all content to the table
    itemValues.put(key, AttributeValue.builder().s(keyVal).build());
    itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
    itemValues.put(albumTitle,
AttributeValue.builder().s(albumTitleValue).build());
    itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

    PutItemRequest request = PutItemRequest.builder()
        .tableName(tableName)
        .item(itemValues)
        .build();

    try {
        ddb.putItem(request);
        System.out.println(tableName + " was successfully updated");

    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be found.
\n", tableName);
        System.err.println("Be sure that it exists and that you've typed its name
correctly!");
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

「」で[完全な例](#)を参照してください [GitHub](#)。

## テーブルの既存の項目の更新

テーブルにすでに存在する項目の属性を更新するには、DynamoDbClient の `updateItem` メソッドを呼び出して、テーブル名、プライマリキーの値、更新するフィールドのマップを渡します。

### Note

アカウントとリージョンの名前付きテーブルが存在しない場合、または渡したプライマリキーで識別される項目が存在しない場合は、[ResourceNotFoundException](#) がスローされます。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.HashMap;
```

## Code

```
public static void updateTableItem(DynamoDbClient ddb,
                                   String tableName,
                                   String key,
                                   String keyVal,
                                   String name,
                                   String updateVal){

    HashMap<String,AttributeValue> itemKey = new HashMap<String,AttributeValue>();

    itemKey.put(key, AttributeValue.builder().s(keyVal).build());

    HashMap<String,AttributeValueUpdate> updatedValues =
        new HashMap<String,AttributeValueUpdate>();

    // Update the column specified by name with updatedVal
    updatedValues.put(name, AttributeValueUpdate.builder()
```

```
        .value(AttributeValue.builder().s(updateVal).build())
        .action(AttributeAction.PUT)
        .build());

UpdateItemRequest request = UpdateItemRequest.builder()
    .tableName(tableName)
    .key(itemKey)
    .attributeUpdates(updatedValues)
    .build();

try {
    ddb.updateItem(request);
} catch (ResourceNotFoundException e) {
    System.err.println(e.getMessage());
    System.exit(1);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

System.out.println("Done!");
}
```

「」で[完全な例](#)を参照してください GitHub。

## テーブルの既存の項目の削除

の `DynamoDbClient.deleteItem` メソッドを使用して、テーブル名とプライマリキーバリューを指定することで、テーブルに存在する項目を削除できます。

### Note

アカウントとリージョンの名前付きテーブルが存在しない場合、または渡したプライマリキーで識別される項目が存在しない場合は、[ResourceNotFoundException](#) がスローされます。

## インポート

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

```
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;
```

## Code

```
public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal) {

    HashMap<String,AttributeValue> keyToGet =
        new HashMap<String,AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal)
        .build());

    DeleteItemRequest deleteReq = DeleteItemRequest.builder()
        .tableName(tableName)
        .key(keyToGet)
        .build();

    try {
        ddb.deleteItem(deleteReq);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

「」で[完全な例](#)を参照してください GitHub。

## 詳細情報

- Amazon DynamoDB デベロッパーガイドの[項目の操作のガイドライン](#)
- Amazon DynamoDB デベロッパーガイドの[DynamoDB の項目の操作](#)

## AWS SDK for Java 2.xを使用して Java オブジェクトを DynamoDB アイテムにマッピングする

[DynamoDB Enhanced Client API](#) は、SDK for Java v1.x の `DynamoDBMapper` クラスの後継となる高レベルのライブラリです。クライアント側のクラスを DynamoDB テーブルにマッピングする簡

単な方法を提供します。テーブルとそれに対応するデータクラス間の関係は、コード内で定義します。関係を定義したら、DynamoDB のテーブルまたは項目に対して、さまざまな作成、読み取り、更新、または削除 (CRUD) オペレーションを直感的に実行できます。

DynamoDB Enhanced Client API には、定義されたスキーマに従わないドキュメントタイプの項目を操作できる [拡張ドキュメント API](#) も含まれています。

DynamoDB Enhanced Client API については、以下のトピックで説明します。

- [DynamoDB Enhanced Client API の使用を開始する](#)
- [DynamoDB Enhanced Client API の基礎について学ぶ](#)
- [高度なマッピング機能を使用する](#)
- [DynamoDB 用の拡張ドキュメント API を使用して JSON ドキュメントを操作する](#)
- [拡張機能を使用する](#)
- [DynamoDB 拡張クライアント API を非同期的に使用する](#)
- [データクラス注釈](#)

## DynamoDB Enhanced Client API の使用を開始する

次のチュートリアルでは、DynamoDB Enhanced Client API を操作するために必要な基本事項を紹介します。

### 依存関係を追加する

プロジェクトで DynamoDB Enhanced Client API を使い始めるには、`dynamodb-enhanced` Maven アーティファクトへの依存関係を追加します。これは次の例で示されます。

### 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>
</project>
```



```
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>dynamodb-enhanced</artifactId>
  </dependency>
</dependencies>
...
</project>
```

Maven central リポジトリで[最新バージョン](#)を検索し、**<VERSION>** をこの値に置き換えます。

## Gradle

```
repositories {
  mavenCentral()
}
dependencies {
  implementation(platform("software.amazon.awssdk:bom:<VERSION>"))
  implementation("software.amazon.awssdk:dynamodb-enhanced")
  ...
}
```

Maven central リポジトリで[最新バージョン](#)を検索し、**<VERSION>** をこの値に置き換えます。

## データクラス `TableSchema` から生成する

[TableSchema](#) を使用すると、拡張クライアントは DynamoDB 属性値をクライアント側クラスとの間でマッピングできるようになります。このチュートリアルでは、静的データクラスから派生し、ビルダーを使用してコードから生成された `TableSchema` について説明します。

## 注釈付きデータクラスを使用する

SDK for Java 2.x には、データクラスと一緒に使用することで、Java Bean で使用して、クラスをテーブルにマッピングするための `TableSchema` をすばやく生成できる [一連の注釈](#) が含まれています。

まず、[JavaBean 仕様](#) に準拠したデータクラスを作成します。この仕様では、クラスには引数のないパブリックコンストラクタが必要であり、クラスの各属性にはゲッターとセッターが必要です。データクラスが `DynamoDbBean` であることを示すクラスレベルの注釈を含めます。また、少なくとも、`getters` と `setters` にはプライマリキー属性の `DynamoDbPartitionKey` 注釈を含めます。

[属性レベルの注釈](#)はゲッターまたはセッターに適用できますが、両方に適用することはできません。

### Note

用語propertyは通常、にカプセル化された値に使用されます JavaBean。ただし、このガイドでは、DynamoDB で使用されている用語との一貫性を保つため、代わりに attribute という用語を使用しています。

次のCustomerクラスは、クラス定義を DynamoDB テーブルにリンクする注釈を示しています。

### Customer クラス

```
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 + "];"
}
}
```

注釈付きデータクラスを作成したら、次のスニペットに示すように、それを使って `TableSchema` を作成します。

```
static final TableSchema<Customer> customerTableSchema =
    TableSchema.fromBean(Customer.class);
```

`TableSchema` は静的で変更不可能であるように設計されています。通常、クラスロード時にインスタンス化できます。

静的 `TableSchema.fromBean()` ファクトリメソッドは、Bean を調べて、DynamoDB 属性との間でデータクラス属性 (プロパティ) のマッピングを生成します。

複数のデータクラスで構成されるデータモデルの操作例については、[???](#) セクションの「Person クラス」を参照してください。

## ビルダーを使用する

テーブルスキーマをコードで定義すれば、Bean イントロスペクションのコストを省くことができます。スキーマをコーディングする場合、クラスは JavaBean 命名基準に従う必要も、注釈を付ける必要もありません。次の例ではビルダーを使用しており、注釈を使用する `Customer` クラスの例と同じです。

```
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();
```

## 拡張クライアントと `DynamoDbTable` を作成する

### 拡張クライアントを作成する

[DynamoDbEnhancedClient](#) クラスまたはその非同期の対応する

は [DynamoDbEnhancedAsyncClient](#)、DynamoDB Enhanced Client API を使用するエントリポイントです。

拡張クライアントでは、作業を実行するための標準の [DynamoDbClient](#) が必要です。API には、`DynamoDbEnhancedClient` インスタンスを作成する 2 つの方法があります。以下のスニペットに示す 1 つ目のオプションは、構成設定から選択したデフォルト設定を使用して標準の `DynamoDbClient` を作成します。

```
DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.create();
```

基礎となる標準クライアントを設定する場合は、次のスニペットに示すように、拡張クライアントのビルダーメソッドにそれを提供することができます。

```
// Configure an instance of the standard DynamoDbClient.
DynamoDbClient standardClient = DynamoDbClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

// Use the configured standard client with the enhanced client.
DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
    .dynamoDbClient(standardClient)
```

```
.build();
```

**DynamoDbTable** インスタンスを作成します。

[DynamoDbTable](#) は、TableSchema によって提供されるマッピング機能を使用する DynamoDB テーブルのクライアント側表現と考えます。DynamoDbTable クラスには、単一の DynamoDB テーブルを操作できる CRUD オペレーションのメソッドが用意されています。

DynamoDbTable<T> は、カスタムクラスでもドキュメントタイプのアイテムを操作するときの EnhancedDocument でも、単一の型引数を取る汎用クラスです。この引数タイプは、使用するクラスと単一の DynamoDB テーブルとの関係を確立します。

次のスニペットに示すように、DynamoDbEnhancedClient の table() ファクトリメソッドを使用して、DynamoDbTable インスタンスを作成します。

```
static final DynamoDbTable<Customer> customerTable =  
    enhancedClient.table("Customer", TableSchema.fromBean(Customer.class));
```

DynamoDbTable インスタンスは不変であり、アプリケーション全体で使用できるので、シングルトンの候補です。

これで、コードで Customer インスタンスを操作できる DynamoDB テーブルのインメモリ表現ができるようになりました。実際の DynamoDB テーブルは、存在する場合と存在しない場合があります。Customer という名前のテーブルがすでに存在する場合は、そのテーブルに対して CRUD オペレーションを実行し始めることができます。テーブルが見つからない場合、次のセクションで説明するように、DynamoDbTable インスタンスを使用してテーブルを作成します。

必要に応じて、DynamoDB テーブルを作成する

DynamoDbTable インスタンスを作成したら、そのインスタンスを使用して DynamoDB でテーブルを 1 回だけ作成します。

テーブル作成例コード

次の例では、Customer データクラスに基づいて DynamoDB テーブルを作成します。

この例では、クラス名と同じ Customer という名前で DynamoDB テーブルを作成しますが、テーブル名は別の名前でもかまいません。テーブルにどのような名前を付けるにしても、テーブルを操作するには他のアプリケーションでもこの名前を使用する必要があります。基になる DynamoDB テー

ブルを操作するために、別の `DynamoDbTable` オブジェクトを作成するときにはいつでも、この名前を `table()` メソッドに指定してください。

`createTable` メソッドに渡される Java lambda パラメータ、`builder`、によって、[テーブルをカスタマイズ](#)できます。この例では、[プロビジョニングされたスループット](#)が設定されています。テーブルの作成時にデフォルト設定を使用する場合は、次のスニペットに示すようにビルダーをスキップします。

```
customerTable.createTable();
```

デフォルト設定を使用すると、[プロビジョニングされたスループット](#)の値は設定されません。代わりに、テーブルの請求モードは[オンデマンド](#)に設定されます。

この例では、レスポンスで受け取ったテーブル名を出力しようとする前に、[DynamoDbWaiter](#) を使用しています。テーブルの作成には少し時間がかかります。したがって、ウェーターを使用すると、テーブルを使用する前に DynamoDB サービスをポーリングしてテーブルが存在するかどうかを確認するロジックを記述する必要がなくなります。

## インポート

```
import com.example.dynamodb.Customer;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

## コード

```
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

DynamoDB テーブルの属性名は、テーブルがデータクラスから生成される場合、小文字で始まります。テーブルの属性名を大文字で始めたい場合は、[@DynamoDbAttribute\(NAME\)](#) [注釈](#)を使用して、必要な名前をパラメータとして指定します。

## オペレーションを実行する

テーブルが作成されたら、`DynamoDbTable` インスタンスを使用して DynamoDB テーブルに対して操作を実行します。

次の例では、[Customer データクラス](#)のインスタンスとともにシングルトン `DynamoDbTable<Customer>` をパラメータとして渡し、テーブルに新しいアイテムを追加します。

```
public static void putItemExample(DynamoDbTable<Customer> customerTable, Customer
customer){
    logger.info(customer.toString());
    customerTable.putItem(customer);
}
```

## Customer オブジェクト

```
Customer customer = new Customer();
customer.setId("1");
customer.setCustName("Customer Name");
customer.setEmail("customer@example.com");
customer.setRegistrationDate(Instant.parse("2023-07-03T10:15:30.00Z"));
```

customer オブジェクトを DynamoDB サービスに送信する前に、オブジェクトの toString() メソッドの出力をログに記録して、拡張クライアントが送信するものと比較します。

```
Customer [id=1, name=Customer Name, email=customer@example.com,
  regDate=2023-07-03T10:15:30Z]
```

ワイヤレベルのログ記録には、生成されたリクエストのペイロードが表示されます。拡張クライアントはデータクラスから低レベルの表現を生成しました。Java の Instant タイプである regDate 属性は、DynamoDB の文字列として表されます。

```
{
  "TableName": "Customer",
  "Item": {
    "registrationDate": {
      "S": "2023-07-03T10:15:30Z"
    },
    "id": {
      "S": "1"
    },
    "custName": {
      "S": "Customer Name"
    },
    "email": {
      "S": "customer@example.com"
    }
  }
}
```

### 既存のテーブルを使用する

前のセクションでは、Java データクラスから DynamoDB テーブルを作成する方法を示しました。既存のテーブルがあって、拡張クライアントの機能を使いたい場合は、そのテーブルを操作する



Java データクラスを作成できます。DynamoDB テーブルを調べ、必要な注釈をデータクラスに追加する必要があります。

既存のテーブルを操作する前に、`DynamoDbEnhanced.table()` メソッドを呼び出します。これは、前の例で次のステートメントを使用して行われました。

```
DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer",  
    TableSchema.fromBean(Customer.class));
```

`DynamoDbTable` インスタンスが返されたら、基になるテーブルですぐに作業を開始できます。`DynamoDbTable.createTable()` メソッドを呼び出してテーブルを再作成する必要はありません。

次の例では、DynamoDB テーブルから `Customer` インスタンスをすぐを取得することで示されます。

```
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()` メソッドで使用されるテーブル名は、既存の DynamoDB テーブル名と一致する必要があります。

## DynamoDB Enhanced Client API の基礎について学ぶ

このトピックでは、DynamoDB Enhanced Client API の基本機能について説明し、[標準の DynamoDB client API](#) と比較します。

DynamoDB Enhanced Client API を初めて使用する場合は、[入門チュートリアル](#)を読んで基本的なクラスに慣れることをおすすめします。

## Java の DynamoDB アイテム

DynamoDB テーブルにはアイテムが保存されます。ユースケースに応じて、Java 側のアイテムは静的に構造化されたデータまたは動的に作成された構造の形をとることができます。

ユースケースで一貫性のある属性セットを持つアイテムが必要な場合は、[注釈付きクラス](#)を使用するか、[ビルダー](#)を使用して適切な静的型 TableSchema を生成します。

あるいは、さまざまな構造から構成されるアイテムを保存する必要がある場合は、DocumentTableSchema を作成します。DocumentTableSchema は[拡張ドキュメント API](#)の一部であり、必要なのは静的型プライマリキーのみで、EnhancedDocument インスタンスと連携してデータ要素を保持します。拡張ドキュメント API については別の[トピック](#)で説明しています。

### データモデルクラスの属性タイプ

DynamoDB は、Java の豊富なタイプシステムに比べて[少数の属性型](#)をサポートしています。DynamoDB Enhanced Client API には Java クラスのメンバーと DynamoDB 属性タイプを相互に変換するメカニズムを提供します。

Java データクラスの属性タイプ (プロパティ) は、プリミティブではなくオブジェクトタイプである必要があります。例えば、Long および int プリミティブではなく、常に long および Integer オブジェクトデータ型を使用します。

デフォルトでは、DynamoDB Enhanced Client API は、整数、[文字列](https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html) <https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html>、インスタント [BigDecimal](#) など、多数のタイプの属性コンバーターをサポートしています。 <https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/enhanced/dynamodb/internal/converter/attribute/InstantAsStringAttributeConverter.html> リストは、[AttributeConverter インターフェイスの既知の実装クラスに表示されます](#)。リストには、マップ、リスト、セットなど、さまざまなタイプとコレクションが含まれています。

デフォルトでサポートされていない、または JavaBean 規則に準拠していない属性タイプのデータを保存するには、変換を実行するカスタム AttributeConverter 実装を記述できます。[例](#)については、「属性変換」セクションを参照してください。

クラスが Java Bean 仕様に準拠する属性型 (または[不変データクラス](#)) のデータを保存するには、2つの方法があります。

- ソースファイルにアクセスできる場合は、クラスに @DynamoDbBean (または @DynamoDbImmutable) という注釈を付けることができます。ネストされた属性について説明しているセクションでは、注釈付きクラスの使用[例](#)を示しています。

- 属性 JavaBean のデータクラスのソースファイルにアクセスできない場合 (またはアクセス可能なクラスのソースファイルに注釈を付けたくない場合)、ビルダーアプローチを使用できます。これにより、キーを定義せずにテーブルスキーマが作成されます。次に、このテーブルスキーマを別のテーブルスキーマ内にネストしてマッピングを実行できます。ネストされた属性セクションには、ネストされたスキーマの[使用例](#)があります。

## Null 値

putItem API を使用する場合、拡張クライアントは、マッピングされたデータオブジェクトの NULL 値属性を DynamoDB へのリクエストに含めません。

updateItem リクエストの場合、NULL 値の属性はデータベース上のアイテムから削除されます。一部の属性値を更新し、他の属性値はそのままにしておきたい場合は、変更してはいけない他の属性の値をコピーするか、更新ビルダーの [ignoreNull \(\)](#) メソッドを使用してください。

次の例は、the updateItem() メソッドの ignoreNulls() を示しています。

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

    // Put item with values for all attributes.
    customerDynamoDbTable.putItem(customer);

    // Create a Customer instance with the same id value, but a different name
    value.
    // Do not set the 'registrationDate' attribute.
    Customer custForUpdate = new Customer();
    custForUpdate.setCustName("NewName");
    custForUpdate.setEmail("email");
    custForUpdate.setId("1");

    // Update item without setting the registrationDate attribute.
    customerDynamoDbTable.updateItem(b -> b
        .item(custForUpdate)
        .ignoreNulls(Boolean.TRUE));

    Customer updatedWithNullsIgnored = customerDynamoDbTable.getItem(customer);
    // registrationDate value is unchanged.
```

```
logger.info(updatedWithNullsIgnored.toString());

customerDynamoDbTable.updateItem(custForUpdate);
Customer updatedWithNulls = customerDynamoDbTable.getItem(customer);
// registrationDate value is null because ignoreNulls() was not used.
logger.info(updatedWithNulls.toString());
}
}

// Logged lines.
Customer [id=1, custName=NewName, email=email,
registrationDate=2023-04-05T16:32:32.056Z]
Customer [id=1, custName=NewName, email=email, registrationDate=null]
```

## DynamoDB Enhanced Client API の基本メソッド

拡張クライアントの基本的なメソッドは、その名前に由来する DynamoDB サービスオペレーションにマッピングされます。次の例は、各方法の最も単純なバリエーションを示しています。拡張リクエストオブジェクトを渡すことで、各メソッドをカスタマイズできます。拡張リクエストオブジェクトは、標準の DynamoDB クライアントで使用できるほとんどの機能を提供します。これらは、AWS SDK for Java 2.x API リファレンスで完全に文書化されています。

この例では前に示した [the section called “Customer クラス”](#) を使用しています。

```
// CreateTable
customerTable.createTable();

// GetItem
Customer customer =
customerTable.getItem(Key.builder().partitionValue("a123").build());

// UpdateItem
Customer updatedCustomer = customerTable.updateItem(customer);

// PutItem
customerTable.putItem(customer);

// DeleteItem
Customer deletedCustomer =
customerTable.deleteItem(Key.builder().partitionValue("a123").sortValue(456).build());

// Query
```

```
PageIterable<Customer> customers = customerTable.query(keyEqualTo(k ->
    k.partitionValue("a123")));

// Scan
PageIterable<Customer> customers = customerTable.scan();

// BatchGetItem
BatchGetResultPageIterable batchResults =
    enhancedClient.batchGetItem(r -> r.addReadBatch(ReadBatch.builder(Customer.class)
        .mappedTableResource(customerTable)
        .addGetItem(key1)
        .addGetItem(key2)
        .addGetItem(key3)
        .build()));

// BatchWriteItem
batchResults = enhancedClient.batchWriteItem(r ->
    r.addWriteBatch(WriteBatch.builder(Customer.class)
        .mappedTableResource(customerTable)
        .addPutItem(customer)
        .addDeleteItem(key1)
        .addDeleteItem(key1)
        .build()));

// TransactGetItems
transactResults = enhancedClient.transactGetItems(r -> r.addGetItem(customerTable,
    key1)
    .addGetItem(customerTable,
    key2));

// TransactWriteItems
enhancedClient.transactWriteItems(r -> r.addConditionCheck(customerTable,
    i -> i.key(orderKey)
    .conditionExpression(conditionExpression))
    .addUpdateItem(customerTable, customer)
    .addDeleteItem(customerTable, key));
```

## DynamoDB Enhanced Client を標準の DynamoDB クライアントと比較する

DynamoDB client API ([標準](#)と[拡張](#)) はどちらも、DynamoDB テーブルを操作して CRUD (作成、読み取り、更新、削除) データレベルの操作を実行できます。クライアント API の違いは、その方法にあります。標準クライアントを使用すると、低レベルのデータ属性を直接操作できます。拡張クライア

ント API では、使い慣れた Java クラスが使用され、バックグラウンドで低レベル API にマッピングされます。

どちらのクライアント API もデータレベルの操作をサポートしていますが、標準の DynamoDB クライアントはリソースレベルの操作もサポートします。リソースレベルのオペレーションは、バックアップの作成、テーブルの一覧表示、テーブルの更新など、データベースを管理します。拡張クライアント API は、テーブルの作成、説明、削除など、特定の数のリソースレベルの操作をサポートします。

2 つのクライアント API が使用するさまざまなアプローチの違いを説明するために、以下のコード例は、標準クライアントと拡張クライアントを使用して同じ ProductCatalog テーブルを作成する方法を示しています。

比較: 標準の DynamoDB クライアントを使用してテーブルを作成

```
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))
);
```

## 比較: DynamoDB Enhanced Client を使用したテーブルの作成

```
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))
    );
```

拡張クライアントは、以下の注釈付きデータクラスを使用します。DynamoDB Enhanced Client は、Java データ型を DynamoDB データ型にマッピングして、より簡潔でわかりやすいコードにします。ProductCatalog は、DynamoDB Enhanced Client で不変クラスを使用する例です。マッピングされたデータクラスでの不変クラスの使用については、このトピックの[後半で説明します](#)。

### ProductCatalog クラス

```
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);
        }
    }
}

```

次の 2 つのバッチ文字起こしのコード例は、拡張クライアントではなく標準クライアントを使用する場合の冗長性とタイプの安全性の欠如を示しています。

比較: 標準の DynamoDB クライアントを使ったバッチ書き込み

```

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());
}
```

## 比較: DynamoDB Enhanced Client を使ったバッチ書き込み

```
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());
}
```

## 不変データクラスでの操作

DynamoDB Enhanced Client API のマッピング機能は、不変データクラスで動作します。不変クラスにはゲッターしかなく、SDK がクラスのインスタンスを作成するために使用するビルダークラスが必要です。不変クラスは、[カスタマークラス](#)に示されている `@DynamoDbBean` 注釈を使用する代わりに、使用するビルダークラスを示すパラメータを受け取る `@DynamoDbImmutable` 注釈を使用します。

次のクラスは `Customer` の不変バージョンです。

```
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); }
}
}
```

データクラスに `@DynamoDbImmutable` 注釈を付けるには、次の要件を満たす必要があります。

1. `Object.class` のオーバーライドされておらず、`@DynamoDbIgnore` 注釈も付いていないすべてのメソッドは、DynamoDB テーブルの属性のゲッターでなければなりません。
2. すべてのゲッターには、ビルダークラスに対応する大文字と小文字を区別するセッターが必要です。
3. 次のコンストラクト条件のうち 1 つだけ満たす必要があります。
  - ビルダークラスにはパブリックデフォルトコンストラクタが必要です。
  - データクラスには、パラメータを取らずにビルダークラスのインスタンスを返す、`builder()` という名前のパブリック静的メソッドが必要です。このオプションは不変 `Customer` クラスに表示されます。
4. ビルダークラスには、パラメータを取らずに不変クラスのインスタンスを返す、`build()` という名前のパブリックメソッドが必要です。

不変クラスの `TableSchema` を作成するには、次のスニペットに示すように `TableSchema` の `fromImmutableClass()` メソッドを使用します。

```
static final TableSchema<CustomerImmutable> customerImmutableTableSchema =
    TableSchema.fromImmutableClass(CustomerImmutable.class);
```

不変クラスから DynamoDB テーブルを作成できるのと同様に、次のスニペットの例に示すように、DynamoDbTable の createTable() を 1 回呼び出すだけで不変クラスからテーブルを作成できます。

```
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));
}
```

Lombok などのサードパーティライブラリを使用します。

[Project Lombok](#) などのサードパーティライブラリは、不変オブジェクトに関連するボイラープレートコードを生成するのに役立ちます。DynamoDB Enhanced Client API は、データクラスがこのセクションで説明する規則に従っている限り、これらのライブラリで動作します。

次の例は、Lombok 注釈付きの不変 CustomerImmutable クラスを示しています。Lombok の onMethod 機能が、@DynamoDbPartitionKey のような属性ベースの DynamoDB 注釈を生成されたコードにコピーすることに注意してください。

```
@Value
@Builder
@dynamoDbImmutable(builder = Customer.CustomerBuilder.class)
public class Customer {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;

    @Getter(onMethod_=@DynamoDbSortKey)
    private String email;

    @Getter(onMethod_=@DynamoDbSecondaryPartitionKey(indexNames = "customers_by_name"))
    private String name;
```

```
@Getter(onMethod_=@DynamoDbSecondarySortKey(indexNames = {"customers_by_date",
"customers_by_name"}))
private Instant createdAt;
}
```

## 式と条件を使用する

DynamoDB Enhanced Client API の式は、[DynamoDB 式](#)の Java の表現です。

DynamoDB Enhanced Client API では、次の 3 種類の式を使用します。

### [Expression](#)

Expression クラスは、条件とフィルタを定義するときに使用されます。

### [QueryConditional](#)

このタイプの式は、クエリオペレーションの[キー条件](#)を表します。

### [UpdateExpression](#)

このクラスは DynamoDB [更新式](#)の記述に役立ち、現在、拡張フレームワークでアイテムを更新する際に使用されています。

## 式の構造分析

式は以下のような構成になっています。

- 文字列式 (必須)。文字列には、属性名と属性値のプレースホルダー名を含む DynamoDB 論理式が含まれています。
- 式値のマッピング (通常は必須)。
- 式名のマッピング (オプション)。

ビルダーを使用して、次のような一般的な形式の Expression オブジェクトを生成します。

```
Expression expression = Expression.builder()
    .expression(<String>)
    .expressionNames(<Map>)
    .expressionValues(<Map>)
    .build()
```

Expression は通常、式値のマップを必要とします。マップは文字列式のプレースホルダーの値を提供します。マップキーはコロン (:) が付いたプレースホルダー名で構成され、マップ値は [AttributeValue](#) のインスタンスです。 [AttributeValues](#) クラスには、リテラルから AttributeValue インスタンスを生成する便利なメソッドがあります。または、 `AttributeValue.Builder` を使用して AttributeValue インスタンスを生成することもできます。

次のスニペットは、コメント行 2 の後に 2 つのエントリがあるマップを示しています。コメント行 1 の後に表示される `expression()` メソッドに渡される文字列には、DynamoDB が操作を実行する前に DynamoDB が解決するプレースホルダーが含まれています。 `price` は許容される属性名であるため、このスニペットには式名 of マップは含まれていません。

```
public static void scanAsync(DynamoDbAsyncTable productCatalog) {
    ScanEnhancedRequest request = ScanEnhancedRequest.builder()
        .consistentRead(true)
        .attributesToProject("id", "title", "authors", "price")
        .filterExpression(Expression.builder()
            // 1. :min_value and :max_value are placeholders for the values
provided by the map
            .expression("price >= :min_value AND price <= :max_value")
            // 2. Two values are needed for the expression and each is
supplied as a map entry.
            .expressionValues(
                Map.of( ":min_value", numberValue(8.00),
                    ":max_value", numberValue(400_000.00)))
            .build())
        .build();
}
```

DynamoDB テーブル内の属性名が予約語か、数値で始まるか、またはスペースを含む場合、Expression には式名 of マップが必要です。

たとえば、属性名が前のコード例で `price` ではなく `1price` だった場合は、次の例のように変更する必要があります。

```
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();
```



```
.build();
```

式名のプレースホルダーはポンド記号 (#) で始まります。式名のマップのエントリでは、プレースホルダーをキー、属性名を値として使用します。マップは `expressionNames()` メソッドで式ビルダーに追加されます。DynamoDB は操作を実行する前に属性名を解決します。

文字列式で関数を使用する場合、式の値は不要です。式関数の例は `attribute_exists(<attribute_name>)` です。

次の例では、[DynamoDB 関数](#) を使用する `Expression` を構築します。この例の式文字列はプレースホルダーを使用していません。この式を `putItem` オペレーションに使用すると、`movie` 属性値がデータオブジェクトの `movie` 属性と等しいアイテムがデータベースにすでに存在するかどうかを確認できます。

```
Expression exp = Expression.builder().expression("attribute_not_exists  
(movie)").build();
```

DynamoDB デベロッパーガイドには、DynamoDB で使用される [低レベルの式](#) に関する完全な情報が記載されています。

## 条件式と条件

`putItem()`、`updateItem()`、`deleteItem()` メソッドを使用する場合、トランザクションオペレーションやバッチオペレーションを使用する場合は、[Expression](#) オブジェクトを使用して、DynamoDB が操作を続行するために満たすべき条件を指定します。これらの式は条件式と呼ばれています。例については、このガイドの [トランザクション例](#) の `addDeleteItem()` メソッド (コメント行 1 の後) で使われている条件式を参照してください。

`query()` メソッドを操作する場合、条件は [QueryConditional](#) として表されます。QueryConditional クラスには、DynamoDB から読み取るアイテムを決定する基準を記述するのに役立つ静的で便利なメソッドがいくつかあります。

QueryConditionals の例については、このガイドの [the section called “Query メソッドの例”](#) セクションの最初のコード例を参照してください。

## フィルタ式

フィルタ式は、スキャンやクエリオペレーションで、返されるアイテムをフィルターするために使用されます。

フィルタ式はデータベースからすべてのデータが読み取られた後に適用されるため、読み取りコストはフィルターがない場合と同じになります。Amazon DynamoDB デベロッパーガイドには、[クエリ](#)オペレーションと[スキャン](#)オペレーションの両方にフィルター式を使用する方法の詳細が記載されています。

次の例は、スキャンリクエストに追加されたフィルタ式を示しています。この条件では、返品されるアイテムは、価格が 8.00 から 80.00 までの価格に限定されます。

```
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();
```

## 更新式

DynamoDB Enhanced Client の `updateItem()` メソッドは、DynamoDB 内のアイテムを更新する標準的な方法を提供します。ただし、より多くの機能が必要な場合は、[UpdateExpressions](#) によって DynamoDB [更新式の構文](#) をタイプセーフに表現します。たとえば、`UpdateExpressions` を使用して、最初に DynamoDB からアイテムを読み取らずに値を増やしたり、リストに個々のメンバーを追加したりできます。現在、更新式は `updateItem()` メソッドのカスタム拡張で使用できます。

更新式の使用例については、このガイドの[カスタム拡張の例](#)を参照してください。

更新式の詳細については、「[Amazon DynamoDB デベロッパーガイド](#)」を参照してください。

### ページ分割された結果を処理する: スキャンとクエリ

DynamoDB Enhanced Client API の `scan`、`query`、および `batch` メソッドは、1 つ以上のページを含むレスポンスを返します。ページには、1 つ以上のアイテムが含まれます。コードはページごとにレスポンスを処理することも、個々のアイテムを処理することもできます。

同期DynamoDbEnhancedClientクライアントによって返されるページ分割レスポンスは [PageIterable](#) オブジェクトを返し、非同期クライアントによって返されるレスポンスは [PagePublisher](#) オブジェクトDynamoDbEnhancedAsyncClientを返します。

このセクションでは、ページ分割された結果の処理について説明し、スキャン API とクエリ API を使用する例を紹介します。

## テーブルのスキャン

SDK [scan](#) のメソッドは、同じ名前の [DynamoDB オペレーション](#)に対応しています。DynamoDB Enhanced Client API にも同じオプションがありますが、使い慣れたオブジェクトモデルを使用してページ分割を処理します。

まず、同期マッピングクラスの `scan`メソッドを調べて、`PageIterable`インターフェイスを調べます[DynamoDbTable](#)。

## 同期 API を使用する

次の例は、[式](#)を使用して返されるアイテムをフィルタリングする `scan` メソッドを示しています。[ProductCatalog](#) は、前に示したモデルオブジェクトです。

コメント行 1 の後に表示されるフィルタリングする式は、返される `ProductCatalog` アイテムを、価格値が 8.00 から 80.00 の間にあるものに限定しています。

この例では、コメント行 2 の後に示されている `attributesToProject` メソッドを使用して `isbn` 値を除外しています。

コメント行 3 では、`scan` メソッドによって`PageIterable` オブジェクト、`pagedResult` が返されます。`PageIterable` の `stream` メソッドは、ページの処理に使用できる [java.util.Stream](#) オブジェクトを返します。この例では、ページ数がカウントされ、ログに記録されます。

コメント行 4 から始まる例では、`ProductCatalog` アイテムへのアクセス方法が 2 種類示されています。コメント行 2a より後のバージョンは、各ページをストリーミングし、各ページのアイテムをソートしてログに記録します。コメント行 2b より後のバージョンは、ページの繰り返し処理をスキップし、アイテムに直接アクセスします。

`PageIterable` インターフェイスには、[java.lang.Iterable](#) と [SdkIterable](#) という 2 つの親インターフェイスがあるため、結果を処理する方法が複数あります。`Iterable` は `forEach`、`iterator`、`spliterator` メソッドを、`SdkIterable` は `stream` メソッドをもたらします。

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

## 非同期 API を使用する

非同期 scan メソッドは結果を PagePublisher オブジェクトとして返します。PagePublisher インターフェースには、レスポンスページの処理に使用できる subscribe メソッドが 2 つあります。subscribe メソッドの一つは、org.reactivestreams.Publisher 親インターフェースか

らのものです。この最初のオプションを使用してページを処理するには、`subscribe` メソッドに [Subscriber](#) インスタンスを渡します。次の最初の例は、`subscribe` メソッドの使用例を示しています。

2 番目の `subscribe` 方法は、[SdkPublisher](#) インターフェイスから取得されます。このバージョンの `subscribe` は、`Subscriber` ではなく [Consumer](#) を受け入れます。この `subscribe` メソッドのバリエーションは、次の 2 番目の例に示されています。

次の例は、前の例で示されたのと同じフィルタ式を使用する非同期バージョンの `scan` メソッドです。

コメント行 3 の後、`DynamoDbAsyncTable.scan` は `PagePublisher` オブジェクトを返します。次の行では、コードによって `org.reactivestreams.Subscriber` インターフェイス、`ProductCatalogSubscriber` のインスタンスが作成され、コメント行 4 の後に `PagePublisher` がサブスクライブされます。

`Subscriber` オブジェクトは、`ProductCatalogSubscriber` クラスの例のコメント行 8 の後に、`onNext` メソッドの各ページから `ProductCatalog` アイテムを収集します。アイテムはプライベート変数 `List` に保存され、`ProductCatalogSubscriber.getSubscribedItems()` メソッドを使用して呼び出し元のコードからアクセスされます。これはコメント行 5 の後に呼び出されます。

リストが取得されると、コードはすべての `ProductCatalog` アイテムを価格順に並べ替え、各アイテムをログに記録します。

`ProductCatalogSubscriber` クラス [CountDownLatch](#) 内のは、コメント行 5 の後に続行する前に、すべての項目がリストに追加されるまで呼び出し元のスレッドをブロックします。

```
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;
    }
}
```

次のスニペット例では、コメント行 6 以降に `Consumer` を受け入れる `PagePublisher.subscribe` メソッドのバージョンを使用しています。Java ラムダパラメータはページを消費し、各アイテムをさらに処理します。この例では、各ページが処理され、各ページのアイテムがソートされてからログに記録されます。

```
// 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 の items メソッドはモデルインスタンスをアンラップして、コードでアイテムを直接処理できるようにします。このアプローチは次のスニペットに示されています。

```
// 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.
```

## テーブルに対してクエリを実行する

DynamoDbTable クラスの [query\(\)](#) メソッドは、プライマリキー値に基づいてアイテムを探します。@DynamoDbPartitionKey 注釈とオプションの @DynamoDbSortKey 注釈を使用して、データクラスのプライマリキーを定義します。

query() メソッドには、指定された値と一致するアイテムを見つけるためのパーティションキー値が必要です。テーブルにソートキーも定義されている場合は、結果を微調整するための追加比較条件として、その値をクエリに追加することができます。

結果の処理以外は、同期バージョンも非同期バージョンも query() の動作は同じです。scan API と同様に、query API は同期呼び出しの場合は PageIterable を返し、非同期呼び出しの場合は PagePublisher を返します。PageIterable と PagePublisher の使用方法については、前回のスキャンセクションで説明しました。

## Query メソッドの例

以下の query() メソッドコード例では MovieActor クラスを使用しています。データクラスは、パーティションキーの **movie** 属性とソートキーの **actor** 属性で構成される複合プライマリキーを定義します。

このクラスは、**acting\_award\_year** という名前のグローバルセカンダリインデックスを使用していることも通知します。インデックスの複合プライマリキーは、パーティションキーの **actingaward** 属性とソートキーの **actingyear** 属性で構成されます。このトピックの後半で、イ



インデックスの作成方法と使用方法を示すときには、**acting\_award\_year** インデックスを参照します。

## MovieActor クラス

```
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);
    }
}
}
```

以下のコード例では、以下の項目に対してクエリを実行します。

### MovieActor テーブル内の項目

```
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'}

```

次のコードでは、2つの[QueryConditional](#)インスタンスを定義します。は、パーティションキーを単独で使用するか、ソートキーと組み合わせてQueryConditionalsを使用して、DynamoDB サービス API の[キー条件式](#)に対応します。この例では、1行目のコメントの後、パーティション値が **movie01** の項目と一致するkeyEqualインスタンスを定義しています。

この例では、コメント行2の後に **actingschoolname** が何もない項目を除外するフィルター式も定義しています。

コメント行3の後、コードが `DynamoDbTable.query()`メソッドに渡す[QueryEnhancedRequest](#)インスタンスの例を示します。このオブジェクトは、SDKがDynamoDBサービスへのリクエストを生成するために使用するキー条件とフィルターを組み合わせたものです。

```

public static void query(DynamoDbTable movieActorTable) {

    // 1. Define a QueryConditional instance to return items matching a partition
    value.
    QueryConditional keyEqual = QueryConditional.keyEqualTo(b ->
b.partitionValue("movie01"));
    // 1a. Define a QueryConditional that adds a sort key criteria to the partition
    value criteria.
    QueryConditional sortGreaterThanOrEqualTo =
QueryConditional.sortGreaterThanOrEqualTo(b ->
b.partitionValue("movie01").sortValue("actor2"));

```

```
// 2. Define a filter expression that filters out items whose attribute value
is null.
final Expression filterOutNoActingschoolname =
Expression.builder().expression("attribute_exists(actingschoolname)").build();

// 3. Build the query request.
QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
    .queryConditional(keyEqual)
    .filterExpression(filterOutNoActingschoolname)
    .build();
// 4. Perform the query.
PageIterable<MovieActor> pagedResults = movieActorTable.query(tableQuery);
logger.info("page count: {}", pagedResults.stream().count()); // Log number of
pages.

pagedResults.items().stream()
    .sorted()
    .forEach(
        item -> logger.info(item.toString()) // Log the sorted list of
items.
    );
```

以下は、メソッドを実行したときの出力です。出力には movie01 の movieName 値を持つ項目が表示され、**null** と等しい actingSchoolName の項目は表示されません。

```
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'}
```

コメント行 3 の後に示した次のクエリリクエストのバリエーションでは、コードは `keyEqual` `QueryConditional` をコメント行 1a の後に定義された `sortGreaterThanOrEqualTo` `QueryConditional` に置き換えます。次のコードではフィルター式も削除されています。

```
QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
    .queryConditional(sortGreaterThanOrEqualTo)
```

このテーブルには複合プライマリキーがあるため、すべての `QueryConditional` インスタンスにパーティションキー値が必要です。 `sort...` で始まる `QueryConditional` メソッドは、ソートキーが必要であることを示します。結果はソートされません。

このクエリでは次の出力が生成されます。このクエリは、 `movie01` と等しい `movieName` 値を持つアイテムを返し、 `actor2` 以上の `actorName` 値のアイテムのみを返します。フィルターが削除されたため、このクエリでは `actingSchoolName` 属性に値がないアイテムが返されます。

```
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'}
```

## バッチオペレーションを実行する

DynamoDB Enhanced Client API には、 [batchGetItem\(\)](#) と [batchWriteItem\(\)](#) の 2 つのバッチメソッドがあります。

### `batchGetItem()` の例

[DynamoDbTable.batchGetItem\(\)](#) メソッドを使用すると、1 回のリクエストで複数のテーブルから最大 100 個の項目を取得できます。次の例では、前に示した [Customer](#) および [MovieActor](#) のデータクラスを使用しています。

1 行目と 2 行目の後の例では、3 行目のコメントの後に `batchGetItem()` メソッドにパラメータとして追加する [ReadBatch](#) オブジェクトを作成します。コメント行 1 の後のコードでは、`Customer` テーブルから読み取るバッチを構築します。コメント行 1a の後のコードは、プライマリキー値を使用して読み取る項目を指定する [GetItemEnhancedRequest](#) ビルダーの使用を示しています。キー値を指定して項目をリクエストするのとは対照的に、コメント行 1b の後に示されているように、データクラスを使用して項目をリクエストできます。SDK はリクエストを送信する前にバックグラウンドでキー値を抽出します。

2a の後の 2 つのステートメントに示されているように、キーベースのアプローチを使用して項目を指定する場合、DynamoDB が [強力な整合性のある読み込み](#) を実行するように指定することもできま

す。consistentRead() メソッドを使用する場合は、同じテーブルのリクエストされた項目すべてにそのメソッドを使用する必要があります。

DynamoDB が検出した項目を取得するには、コメント行 4 の後に表示される [resultsForTable\(\)](#) メソッドを使用します。リクエストで読み取られた各テーブルのメソッドを呼び出します。resultsForTable() は見つかった項目の一覧を返し、どの java.util.List メソッドでも処理できます。この例では各項目を記録しています。

DynamoDB が処理しなかった項目を見つけるには、コメント行 5 の後の方法を使用します。BatchGetResultPage クラスには、未処理の各キーにアクセスできる [unprocessedKeysForTable\(\)](#) メソッドがあります。[BatchGetItem API リファレンス](#)には、未処理の項目が発生する状況に関する詳細情報が記載されています。

```
public static void batchGetItemExample(DynamoDbEnhancedClient enhancedClient,
                                       DynamoDbTable<Customer> customerTable,
                                       DynamoDbTable<MovieActor> movieActorTable) {

    Customer customer2 = new Customer();
    customer2.setId("2");
    customer2.setEmail("cust2@example.org");

    // 1. Build a batch to read from the Customer table.
    ReadBatch customerBatch = ReadBatch.builder(Customer.class)
        .mappedTableResource(customerTable)
        // 1a. Specify the primary key values for the item.
        .addGetItem(b -> b.key(k ->
k.partitionValue("1").sortValue("cust1@orgname.org")))
        // 1b. Alternatively, supply a data class instances to provide the
primary key values.
        .addGetItem(customer2)
        .build();

    // 2. Build a batch to read from the MovieActor table.
    ReadBatch movieActorBatch = 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()));
});
}
```

サンプルコードを実行する前に、2つのテーブルに次の項目が含まれていると仮定します。

### テーブル内の項目

```
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'}
```

次の出力は、コメント行 4 の後に返され、記録された項目を示しています。

```
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() の例

batchWriteItem() メソッドが 1 つ以上のテーブルに 1 つ以上の項目を入力または削除します。リクエストでは、最大 25 件の個別の入力または削除操作を指定できます。次の例では、前に示した [ProductCatalog](#) または [MovieActor](#) のモデルクラスを使用しています。

WriteBatch オブジェクトはコメント行 1 と 2 の後に作成されます。ProductCatalog テーブルでは、コードによって 1 つの項目が入力され、1 つの項目が削除されます。コメント 2 行目以降の MovieActor テーブルでは、コードによって 2 つの項目が入力され、1 つの項目が削除されます。

batchWriteItem メソッドはコメント行 3 の後に呼び出されます。[builder](#) パラメータは各テーブルのバッチリクエストを提供します。

返された [BatchWriteResult](#) オブジェクトには、未処理のリクエストを表示するための個別のメソッドが操作ごとに用意されています。コメント行 4a の後のコードは未処理の削除リクエストのキーを提供し、コメント行 4b の後のコードは未処理の入力項目を示します。

```
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()));
    }
}
}

```

以下のヘルパーメソッドは入力操作と削除操作のモデルオブジェクトを提供します。

## ヘルパーメソッド

```

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;
    }
}
```

サンプルコードを実行する前に、テーブルに次の項目が含まれていると仮定します。

```
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}
```

サンプルコードが完了すると、テーブルには以下の項目が含まれます。

```
MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='null'}
MovieActor{movieName='Everything Everywhere All at Once', actorName='Michelle Yeoh', actingAward='Best Actress', actingYear=2023, actingSchoolName='Royal Academy of Dance'}
ProductCatalog{id=2, title='Title 55', isbn='1-565-85698', authors=[a, b], price=30.22}
```

MovieActor テーブルでは、Blue Jasmine ムービーアイテムが、`getMovieActorBlanchettPartial()` ヘルパーメソッドを通じて取得された入力リクエストで使用されたアイテムに置き換えられていることに注意してください。データ Bean 属性値が指定されなかった場合、データベース内の値は削除されます。これが、`actingSchoolName` で Blue Jasmine ムービーアイテムの結果が NULL になる理由です。

### Note

API ドキュメントでは、条件式を使用でき、消費された容量とコレクションのメトリクスは個別のリクエストとリクエストで返すことができると記載されていますが、バッチ書き込みシナリオではそうではありません。バッチ操作のパフォーマンスを向上させるため、これらの個々のオプションは無視されます。

## トランザクションオペレーションを実行する

DynamoDB 拡張クライアント API には、`transactGetItems()` および `transactWriteItems()` メソッドが用意されています。SDK for Java のトランザクションによって DynamoDB テーブルに不可分性、一貫性、分離性、耐久性 (ACID) が実現されるため、アプリケーション内でのデータの精度を維持することができます。

### `transactGetItems()` の例

[transactGetItems\(\)](#) メソッドは、項目に対する個別のリクエストを最大 100 件受け付けます。すべてのアイテムは 1 つの不可分トランザクションで読み取られます。Amazon

DynamoDB デベロッパーガイドには、[transactGetItems\(\) メソッドが失敗する原因となる条件](#)や、[transactGetItem\(\)](#) の呼び出し時に使用される分離レベルに関する情報が記載されています。

次の例の 1 行目のコメントのあと、コードは [builder](#) パラメータを使用して [transactGetItems\(\)](#) メソッドを呼び出します。ビルダー [addGetItem\(\)](#) は、SDK が最終リクエストの生成に使用するキー値を含むデータオブジェクトを使用して 3 回呼び出されます。

このリクエストは、コメント行 2 の後に [Document](#) オブジェクトのリストを返します。返されるドキュメントのリストには、アイテムデータの null 以外の [ドキュメント](#) インスタンスがリクエストと同じ順序で含まれています。[Document.getItem\(MappedTableResource<T> mappedTableResource\)](#) メソッドは、アイテムデータが返された場合は型指定されていない [Document](#) オブジェクトを型付きの Java オブジェクトに変換し、それ以外の場合は null を返します。

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

コード例が実行される前は、DynamoDB テーブルには次の項目が含まれています。

```
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'}
```

次の出力が記録されます。リクエストされたアイテムが見つからなかった場合、Blue Jasmine という名前の映画のリクエストの場合とは異なり、アイテムは返されません。

```
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() の例

[transactWriteItems\(\)](#) は、複数のテーブルにわたる 1 回のアトミックトランザクションで最大 100 件の入力、更新、削除アクションを受け付けます。Amazon DynamoDB デベロッパーガイドには、[基盤となる DynamoDB サービスオペレーション](#)の制限と障害条件に関する詳細が記載されています。

### 基本的な の例

次の例では、2 つのテーブルに対して 4 つのオペレーションがリクエストされています。対応するモデルクラスは、前に示した [ProductCatalog](#) および [MovieActor](#) です。

入力、更新、削除の 3 つの操作では、それぞれ専用のリクエストパラメータを使用して詳細を指定します。

コメント行 1 の後のコードは、`addPutItem()` メソッドの単純なバリエーションを示しています。このメソッドは、[MappedTableResource](#) オブジェクトと入力するデータオブジェクトインスタンスを受け入れます。2 行目のコメントの後のステートメントには、[TransactPutItemEnhancedRequest](#) インスタンスを受け付けるバリエーションが表示され

ます。このバリエーションでは、条件式などのオプションをリクエストに追加できます。次の例では、個々のオペレーションの条件式を示しています。

更新操作はコメント行 3 の後に要求されます。[TransactUpdateItemEnhancedRequest](#) には、SDK がモデルオブジェクトの null 値を使って何をするかを設定できる `ignoreNulls()` メソッドがあります。`ignoreNulls()` メソッドが true を返す場合、SDK は null であるデータオブジェクト属性のテーブルの属性値を削除しません。`ignoreNulls()` メソッドが false を返す場合、SDK は DynamoDB サービスにテーブル内の項目から属性を削除するようリクエストします。`ignoreNulls` のデフォルト値は false です。

コメント 4 行目の後のステートメントは、データオブジェクトを取得する削除リクエストのバリエーションを示しています。拡張クライアントは、最後のリクエストをディスパッチする前にキー値を抽出します。

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

以下のヘルパーメソッドは、add\*Item パラメータのデータオブジェクトを提供します。

## ヘルパーメソッド

```
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;
}
```

コード例が実行される前は、DynamoDB テーブルには次の項目が含まれています。



```
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'}
```

コードの実行が完了すると、次の項目がテーブルに表示されます。

```
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'}
```

2 行目の項目は削除され、3 行目と 5 行目には追加された項目が表示されます。4 行目は 1 行目の更新を示しています。アイテム上で変更されたのは price の値だけです。ignoreNulls() が false を返した場合、4 行目は次の行のようになります。

```
ProductCatalog{id=4, title='Title 1', isbn='null', authors=null, price=40.0}
```

## コンディションチェックの例

次の例は、条件チェックの使用を示しています。コンディションチェックは、アイテムの存在を確認したり、データベース内のアイテムの特定の属性の状態をチェックしたりするために使用されます。コンディションチェックでチェックされたアイテムは、トランザクション内の別のオペレーションでは使用できません。

### Note

同じトランザクション内の複数のオペレーションが同じ項目をターゲットとすることはできません。たとえば、同じトランザクション内で同じ項目に対してコンディションチェックと更新を実行することはできません。

この例では、トランザクションによる項目書き込みリクエストにおける各タイプの操作を 1 つずつ示しています。2 行目のコメントの後、addConditionCheck() メソッドは、conditionExpression パラメータが false と評価された場合にトランザクションが失敗する条件を指定します。ヘルパーメソッドブロックに表示されているメソッドから返される条件式は、ムービーの受賞年度 Sophie's Choice が 1982 と等しくないかどうかをチェックします。一致した場合、式は false と評価され、トランザクションは失敗します。

このガイドでは、別のトピックで[式](#)について詳しく説明します。

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

```
    }  
}
```

前のコード例では以下のヘルパーメソッドが使用されています。

## ヘルパーメソッド

```
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;  
}
```

```
}
```

コード例が実行される前は、DynamoDB テーブルには次の項目が含まれています。

```
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'}
```

コードの実行が完了すると、次の項目がテーブルに表示されます。

```
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'}
```

トランザクションが失敗したため、テーブル内の項目は変更されません。ムービー Sophie's Choice の actingYear 値は 1982 であり、transactWriteItem() メソッドが呼び出される前のテーブル内の項目の 2 行目に示されているとおりです。

トランザクションのキャンセル情報を取得するには、transactWriteItems() メソッド呼び出しを try ブロックし、[TransactionCanceledException](#) を catch します。この例の 4 行目のコメントの後、コードは各 [CancellationReason](#) オブジェクトをログに記録します。この例の 3 行目のコメントに続くコードでは、トランザクションが失敗した原因となった項目の値を返すように指定していたため、ログには Sophie's Choice ムービー項目の未加工のデータベース値が表示されます。

```
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.)
```

## 単一オペレーション条件の例

次の例は、トランザクションリクエスト内の 1 つのオペレーションで条件を使用する方法を示しています。コメント行 1 の後の削除操作には、操作の対象項目の値をデータベースと照合する条件が含まれています。この例では、ヘルパーメソッドでコメント行 2 の後に作成した条件式は、映画の制作年が 2013 年と等しくない場合はその項目をデータベースから削除するように指定しています。

[式](#)についてはこのガイドの後半で説明します。

```
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();
}
```

前のコード例では以下のヘルパーメソッドが使用されています。

### ヘルパーメソッド

```
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;
}
```

コード例が実行される前は、DynamoDB テーブルには次の項目が含まれています。

```
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'}
```

コードの実行が完了すると、次の項目がテーブルに表示されます。

```
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'}
```

トランザクションが失敗したため、テーブル内の項目は変更されません。ムービー Blue Jasmine の actingYear 値は、コード例が実行される前の項目リストの 2 行目に表示されている 2013 です。

次の行がコンソールに記録されます。

```
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)
```

## セカンダリインデックスを使用する

セカンダリインデックスは、クエリやスキャンの操作に使用する代替キーを定義することで、データアクセスを向上させます。グローバルセカンダリインデックス (GSI) はベーステーブルのものとは異なる可能性があるパーティションキーおよびソートキーを持ちます。対照的に、ローカルセカンダリインデックス (LSI) はプライマリインデックスのパーティションキーを使用します。

セカンダリインデックス注釈でデータクラスに注釈を付けます。

セカンダリインデックスに含まれる属性には、@DynamoDbSecondaryPartitionKey または @DynamoDbSecondarySortKey 注釈が必要です。

次のクラスは 2 つのインデックスの注釈を表示します。という名前の GSI SubjectLastPostedDateIndex は、パーティションキーに Subject 属性を使用し、

ソートキーLastPostedDateTimeに を使用します。という名前の LSI は、パーティションキーForumNameとして を使用し、ソートキーLastPostedDateTimeとして ForumLastPostedDateIndexを使用します。

Subject 属性には 2 つの役割があることに注意してください。これは、プライマリキーのソートキーであり、 という名前の GSI のパーティションキーですSubjectLastPostedDateIndex。

## MessageThread クラス

MessageThread クラスは、Amazon DynamoDB デベロッパーガイドの[サンプルスレッドテーブル](#)のデータクラスとして使用するのに適しています。

### インポート

```
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 +
        '}';
}
}

```

## インデックスを作成する

SDK for Java のバージョン 2.20.86 以降では、`createTable()` メソッドはデータクラス注釈からセカンダリインデックスを自動的に生成します。デフォルトでは、ベーステーブルのすべての属性がインデックスにコピーされ、プロビジョニングされるスループット値は 20 読み取りキャパシティーユニットと 20 書き込みキャパシティーユニットです。

ただし、2.20.86 より前のバージョンの SDK を使用する場合は、次の例のようにテーブルと一緒にインデックスを作成する必要があります。この例では、Thread テーブルの 2 つのインデックスを構築します。[builder](#) パラメータには、1 行目と 2 行目のコメント行の後に示されているように、両方のタイプのインデックスを設定するメソッドがあります。インデックスビルダーの `indexName()` メソッドを使用して、データクラス注釈で指定されているインデックス名を目的のインデックスタイプに関連付けます。

このコードでは、すべてのテーブル属性がコメント行 3 と 4 のコメント行の後に両方のインデックスに含まれるように構成しています。[属性プロジェクション](#)の詳細については、「Amazon DynamoDB デベロッパーガイド」を参照してください。

```

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))
        )
    );
```

インデックスを使用してクエリを実行する

次の例では、ローカルセカンダリインデックス をクエリしますForumLastPostedDateIndex。

コメント行 2 に続いて、[DynamoDbIndex.query\(\)](#) メソッドを呼び出すときに必要な[QueryConditional](#)オブジェクトを作成します。

コメント 3 行目の後にインデックスの名前を渡すと、クエリするインデックスへの参照が得られます。4 行目のコメントに続いて、QueryConditional オブジェクトに渡されるインデックスの query() メソッドを呼び出します。

また、5 行目のコメントの後に示される 3 つの属性値を返すようにクエリを構成します。attributesToProject() が呼び出されない場合、クエリはすべての属性値を返します。指定した属性名は小文字で始まることに注意してください。これらの属性名は表で使用されているものと一致し、必ずしもデータクラスの属性名と一致するわけではありません。

6 行目のコメントに続いて、結果を反復処理し、クエリによって返される各項目をログ記録し、それをリストに保存して呼び出し元に返します。

```
public static List<MessageThread> queryUsingSecondaryIndices(DynamoDbEnhancedClient
    enhancedClient,
                                                                String lastPostedDate,
                                                                DynamoDbTable<MessageThread> threadTable) {
    // 1. Log the parameter value.
    logger.info("lastPostedDate value: {}", lastPostedDate);

    // 2. Create a QueryConditional whose sort key value must be greater than or
    equal to the parameter value.
    QueryConditional queryConditional =
    QueryConditional.sortGreaterThanOrEqualTo(qc ->
        qc.partitionValue("Forum02").sortValue(lastPostedDate));

    // 3. Specify the index name to query the DynamoDbIndex instance.
```

```
final DynamoDbIndex<MessageThread> forumLastPostedDateIndex =
threadTable.index("ForumLastPostedDateIndex");

// 4. Perform the query by using the QueryConditional object.
final SdkIterable<Page<MessageThread>> pagedResult =
forumLastPostedDateIndex.query(q -> q
    .queryConditional(queryConditional)
    // 5. Request three attribute in the results.
    .attributesToProject("forumName", "subject", "lastPostedDateTime"));

List<MessageThread> collectedItems = new ArrayList<>();
// 6. Iterate through the pages response and sort the items.
pagedResult.stream().forEach(page -> page.items().stream()

.sorted(Comparator.comparing(MessageThread::getLastPostedDateTime))
    .forEach(mt -> {
        // 7. Log the returned items and add the collection to
return to the caller.
        logger.info(mt.toString());
        collectedItems.add(mt);
    }));
return collectedItems;
}
```

クエリが実行される前は、次の項目がデータベースに存在しています。

```
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}
```

1 行目と 6 行目のログ記録ステートメントは、次のようなコンソール出力になります。

```
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}
```

このクエリでは、Forum02 の forumName 値と 2023.03.31 以上の lastPostedDateTime 値の項目が返されました。インデックスには message 属性に値が含まれていますが、結果には空の文字列を含む message 値が表示されます。これは、メッセージ属性がコメント 5 行目以降に投影されなかったためです。

## 高度なマッピング機能を使用する

DynamoDB 拡張クライアント API の高度なテーブルスキーマ機能について説明します。

### テーブルのスキーマタイプを理解する

[TableSchema](#) は DynamoDB 拡張クライアント API のマッピング機能へのインターフェイスです。このマップとの間でデータオブジェクトをマッピングできます [AttributeValues](#)。TableSchema オブジェクトはマッピングするテーブルの構造を知る必要があります。この構造情報は [TableMetadata](#) オブジェクトに格納されます。

拡張クライアント API には、以下のようないくつかの `TableSchema` の実装があります。

注釈付きのクラスから生成されたテーブルスキーマ

注釈付きクラスから `TableSchema` を構築するのは比較的成本のかかる操作であり、アプリケーションの起動時に一度実行することをお勧めします。

### [BeanTableSchema](#)

この実装は Bean クラスの属性と注釈に基づいて構築されています。この方法の例は、「[はじめに](#)」[セクション](#)で説明されています。

#### Note

`BeanTableSchema` が期待どおりに動作しない場合は、`software.amazon.awssdk.enhanced.dynamodb.beans` のデバッグログ記録を有効にします。

### [ImmutableTableSchema](#)

この実装は不変データクラスから構築されています。このアプローチは [??? セクション](#)で説明されています。

ビルダーで生成されたテーブルスキーマ

以下の `TableSchema` は、ビルダーを使用してコードから構築されています。このアプローチは、注釈付きのデータクラスを使用するアプローチよりもコストが低くなります。ビルダーアプローチは注釈の使用を避け、JavaBean 命名基準を必要としません。

### [StaticTableSchema](#)

この実装は可変データクラス用に構築されています。このガイドの「[はじめに](#)」[セクション](#)では、[ビルダーを使用して StaticTableSchema を生成する](#)方法を説明しました。

### [StaticImmutableTableSchema](#)

`StaticTableSchema` を構築する場合と同様に、不変データクラスで使用する [ビルダー](#)を使用してこのタイプ `TableSchema` の実装を生成します。

## 固定スキーマのないデータ用のテーブルスキーマ

### [DocumentTableSchema](#)

TableSchema の他の実装とは異なり、DocumentTableSchema インスタンスには属性を定義しません。通常は、プライマリキーと属性コンバータープロバイダーのみを指定します。EnhancedDocument インスタンスは、個々の要素または JSON 文字列から構築した属性を提供します。

属性を明示的に含めたり除外したりします。

DynamoDB 拡張クライアント API には、データクラス属性がテーブルの属性にならないようにする注釈が用意されています。この API では、データクラスの属性名とは異なる属性名を使用することもできます。

#### 属性を除外する

DynamoDB テーブルにマッピングしてはいけない属性を無視するには、その属性に `@DynamoDbIgnore` 注釈を付けます。

```
private String internalKey;

@DynamoDbIgnore
public String getInternalKey() { return this.internalKey; }
public void setInternalKey(String internalKey) { return this.internalKey =
    internalKey;}
```

#### 属性を含める

DynamoDB テーブルで使用される属性の名前を変更するには、`@DynamoDbAttribute` 注釈を付けて別の名前を指定します。

```
private String internalKey;

@DynamoDbAttribute("renamedInternalKey")
public String getInternalKey() { return this.internalKey; }
public void setInternalKey(String internalKey) { return this.internalKey =
    internalKey;}
```



## 属性変換を制御する

デフォルトでは、テーブルスキーマは、[AttributeConverterProvider](#) インターフェイスのデフォルト実装を通じて多くの一般的な Java タイプのコンバーターを提供します。全体的なデフォルト動作は、カスタム `AttributeConverterProvider` 実装で変更できます。また、1 つの属性のコンバーターを変更することもできます。

使用可能なコンバーターのリストについては、[AttributeConverter](#) 「インターフェイス Java ドキュメント」を参照してください。

### カスタム属性コンバータープロバイダーを提供する

`@DynamoDbBean` (`converterProviders = {...}`) 注釈を使用して、単一の `AttributeConverterProvider` または順序付けられた `AttributeConverterProvider` のチェーンを提供することができます。どのようなカスタム `AttributeConverterProvider` でも `AttributeConverterProvider` インターフェイスを拡張する必要があります。

独自の属性コンバータープロバイダーチェーンを指定すると、デフォルトのコンバータープロバイダー、`DefaultAttributeConverterProvider` がオーバーライドされることに注意してください。`DefaultAttributeConverterProvider` の機能を使用するには、チェーンに組み込む必要があります。

Bean に空の配列 `{}` に注釈を付けすることもできます。これにより、デフォルトを含むすべての属性コンバータープロバイダーの使用が無効になります。この場合、マップされるすべての属性には独自の属性コンバーターが必要です。

次のスニペットは、単一のコンバータープロバイダーを示しています。

```
@DynamoDbBean(converterProviders = ConverterProvider1.class)
public class Customer {

}
```

次のスニペットは、コンバータープロバイダーのチェーンの使用方法を示しています。SDK デフォルトは最後に指定されるため、優先度は最も低くなります。

```
@DynamoDbBean(converterProviders = {
    ConverterProvider1.class,
    ConverterProvider2.class,
    DefaultAttributeConverterProvider.class})
```

```
public class Customer {  
  
}
```

静的テーブルスキーマビルダーにも同じように機能する `attributeConverterProviders()` メソッドがあります。これは次のスニペットに示されています。

```
private static final StaticTableSchema<Customer> CUSTOMER_TABLE_SCHEMA =  
    StaticTableSchema.builder(Customer.class)  
        .newItemSupplier(Customer::new)  
        .addAttribute(String.class, a -> a.name("name")  
            .getter(Customer::getName)  
            .setter(Customer::setName))  
        .attributeConverterProviders(converterProvider1, converterProvider2)  
        .build();
```

### 単一の属性のマッピングをオーバーライドする

単一の属性のマッピング方法をオーバーライドするには、その属性に `AttributeConverter` を指定します。この追加は、テーブルスキーマの `AttributeConverterProviders` で提供されるコンバーターよりも優先されます。これにより、その属性専用のカスタムコンバーターが追加されます。他の属性は、同じタイプの属性であっても、他の属性に明示的に指定されていない限り、そのコンバーターを使用しません。

`@DynamoDbConvertedBy` 注釈は、次のスニペットに示すようにカスタム `AttributeConverter` クラスを指定するために使用されます。

```
@DynamoDbBean  
public class Customer {  
    private String name;  
  
    @DynamoDbConvertedBy(CustomAttributeConverter.class)  
    public String getName() { return this.name; }  
    public void setName(String name) { this.name = name;}  
}
```

静的スキーマのビルダーには、同等の属性ビルダー `attributeConverter()` メソッドがあります。このメソッドは、次に示すように、`AttributeConverter` のインスタンスを取得します。

```
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();
```

## 例

この例は、[java.net.HttpCookie](https://docs.aws.amazon.com/sdk-for-java/v2/developer-guide/cookies.html) オブジェクトの属性コンバーターを提供する `AttributeConverterProvider` 実装を示しています。

次の `SimpleUser` クラスには、`HttpCookie` のインスタンスである `lastUsedCookie` という名前の属性が含まれています。

`@DynamoDbBean` 注釈のパラメータには、コンバーターを提供する 2 つの `AttributeConverterProvider` クラスがリストされています。

## 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();
```

次の例の `CookieConverterProvider` では、`HttpCookieConverter` のインスタンスを提供しています。

```
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);
    }
}
```

## 変換コード

次の `HttpCookieConverter` クラスの `transformFrom()` メソッドでは、コードは `HttpCookie` インスタンスを受け取り、属性として保存される DynamoDB マップに変換します。

`transformTo()` メソッドは DynamoDB マップパラメータを受け取り、名前と値を必要とする `HttpCookie` コンストラクターを呼び出します。

```
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;
    }
}
```

## 属性の更新動作を変更する

更新オペレーションを実行するときに、個々の属性の更新動作をカスタマイズできます。DynamoDB Enhanced Client API の更新オペレーションの例には、[updateItem\(\)](#) と [transactWriteItems\(\)](#) があります。

たとえば、作成した日付のタイムスタンプをレコードに保存したいとします。ただし、データベースにその属性の既存の値がない場合にのみ、その値を書き込むようにしたいとします。この場合、[WRITE\\_IF\\_NOT\\_EXISTS](#) 更新動作を使用します。

次の例は、`createdOn` 属性に動作を追加する注釈を示しています。

```
@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; }
}
```

次の例のコメント行 1 の後のように、静的テーブルスキーマを構築する場合にも同じ更新動作を宣言できます。

```
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();
```

## 他のクラスの属性をフラット化する

テーブルの属性が継承または構成によって複数の異なる Java クラスに分散している場合、DynamoDB Enhanced Client API では属性を 1 つのクラスにフラット化するためのサポートを提供します。

## 継承を使用する

クラスで継承を使用する場合は、以下の方法で階層をフラット化してください。

## 注釈の付いた Bean を使用する

注釈手法では、両方のクラスに `@DynamoDbBean` 注釈が必要で、1 つのクラスに 1 つ以上のプライマリキーの注釈を付ける必要があります。

継承関係を持つデータクラスの例を以下に示します。

## Standard data class

```
@DynamoDbBean
public class Customer extends GenericRecord {
    private String name;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}

@dynamoDbBean
public abstract class GenericRecord {
    private String id;
    private String createdAt;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
        createdAt; }
}
```

## Lombok

Lombok [onMethod のオプション](#)では、@DynamoDbPartitionKey などの属性ベースの DynamoDB 注釈が生成コードにコピーされます。

```
@DynamoDbBean
@Data
@ToString(callSuper = true)
public class Customer extends GenericRecord {
    private String name;
}

@Data
@dynamoDbBean
public abstract class GenericRecord {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;
    private String createdAt;
}
```

### 静的スキーマを使用する

静的スキーマアプローチでは、ビルダーの extend() メソッドを使用して親クラスの属性を子クラスに集約します。これは次の例のコメント行 1 の後で示されています。

```
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::getCreatedAt)

    .setter(org.example.tests.model.inheritance.stat.GenericRecord::setCreatedAt))
    .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();
```

先ほどの静的スキーマの例では、次のデータクラスを使用しています。マッピングは静的テーブルスキーマの構築時に定義されるため、データクラスには注釈は必要ありません。

## データクラス

### Standard data class

```
public class Customer extends GenericRecord {
    private String name;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}

public abstract class GenericRecord {
    private String id;
    private String createdAt;

    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
createdAt; }
```

## Lombok

```
@Data
@ToString(callSuper = true)
public class Customer extends GenericRecord{
    private String name;
}

@Data
public abstract class GenericRecord {
    private String id;
    private String createdAt;
}
```

### 構成を使用する

クラスで構成を使用する場合は、以下の方法で階層をフラット化してください。

### 注釈の付いた Bean を使用する

@DynamoDbFlatten 注釈は格納されているクラスをフラット化します。

以下のデータクラスの例では、@DynamoDbFlatten 注釈を使用して、含まれている GenericRecord クラスのすべての属性を Customer クラスに効果的に追加しています。

### Standard data class

```
@DynamoDbBean
public class Customer {
    private String name;
    private GenericRecord record;

    public String getName() { return this.name; }
    public void setName(String name) { this.name = name; }

    @DynamoDbFlatten
    public GenericRecord getRecord() { return this.record; }
    public void setRecord(GenericRecord record) { this.record = record; }

    @DynamoDbBean
    public class GenericRecord {
        private String id;
```

```

private String createdAt;

@DynamoDbPartitionKey
public String getId() { return this.id; }
public void setId(String id) { this.id = id; }

public String getCreatedAt() { return this.createdAt; }
public void setCreatedAt(String createdAt) { this.createdAt =
createdAt; }
}

```

## Lombok

```

@Data
@DynamoDbBean
public class Customer {
    private String name;
    @Getter(onMethod_=@DynamoDbFlatten)
    private GenericRecord record;
}

@Data
@DynamoDbBean
public class GenericRecord {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;
    private String createdAt;
}

```

フラット化された注釈を使用すると、対象となるさまざまなクラスを必要なだけフラット化できます。以下の制約が適用されます。

- フラット化の後には、すべての属性名が一意でなければなりません。
- パーティションキー、ソートキー、またはテーブル名は複数あってはなりません。

### 静的スキーマを使用する

静的テーブルスキーマを構築するときは、ビルダーの `flatten()` メソッドを使用します。また、含まれているクラスを識別する `getter` メソッドと `setter` メソッドも指定します。

```
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();
```

先ほどの静的スキーマの例では、次のデータクラスを使用しています。

## データクラス

### 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;
}

```

ビルダーパターンを使用して、対象となるさまざまなクラスを必要なだけフラット化できます。

### 他のコードへの影響

@DynamoDbFlatten 属性 (または flatten() ビルダーメソッド) を使用する場合、DynamoDB の項目には、構成オブジェクトの各属性の属性が含まれます。また、構成オブジェクトの属性も含まれます。

これとは対照的に、データクラスに構成クラスの注釈を付け、@DynamoDbFlatten を使用しない場合、その項目は構成オブジェクトとともに単一の属性として保存されます。

たとえば、[構成例によるフラット化](#)で示された Customer クラスを、record 属性をフラット化する場合とフラット化しない場合で比較します。次の表に示すように、JSON との違いを視覚化できます。

フラット化あり	フラット化なし
3つの属性	2つの属性

フラット化あり	フラット化なし
<pre>{   "id": "1",   "createdDate": "today",   "name": "my name" }</pre>	<pre>{   "id": "1",   "record": {     "createdDate": "today",     "name": "my name"   } }</pre>

他のコードが DynamoDB テーブルにアクセスしていて、特定の属性を見つけることを想定している場合には、この違いが重要になります。

### ネストされた属性の使用

DynamoDB のネストされた属性は、別の属性に埋め込まれています。例としては、リスト要素やマップエントリがあります。

Java では、DynamoDB のネストされた属性は、List または Map のクラスのメンバーに対応します。また、次の Person クラスで使用されている Address や PhoneNumber などの複合型のインスタンスにも対応しています。

### Person クラス

```
@DynamoDbBean
public class Person {
    Integer id;
    String firstName;
    String lastName;
    Integer age;
    Map<String, Address> addresses;
    List<PhoneNumber> phoneNumbers;

    List<String> hobbies;

    @DynamoDbPartitionKey
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
```

```
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    public Map<String, Address> getAddresses() {
        return addresses;
    }

    public void setAddresses(Map<String, Address> addresses) {
        this.addresses = addresses;
    }

    public List<PhoneNumber> getPhoneNumbers() {
        return phoneNumbers;
    }

    public void setPhoneNumbers(List<PhoneNumber> phoneNumbers) {
        this.phoneNumbers = phoneNumbers;
    }

    public List<String> getHobbies() {
```

```
        return hobbies;
    }

    public void setHobbies(List<String> hobbies) {
        this.hobbies = hobbies;
    }

    @Override
    public String toString() {
        return "Person{" +
            "id=" + id +
            ", firstName='" + firstName + '\'' +
            ", lastName='" + lastName + '\'' +
            ", age=" + age +
            ", addresses=" + addresses +
            ", phoneNumbers=" + phoneNumbers +
            ", hobbies=" + hobbies +
            '}';
    }
}
```

## Address クラス

```
@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 クラス

```
@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 + '\'' +  
            '}';  
    }  
}
```

### ネストされた属性をマップする

#### 注釈付きのクラスを使用する

カスタムクラスのネストされた属性は、注釈を付けることで保存できます。前に示した Address クラスと PhoneNumber クラスには、@DynamoDbBean 注釈のみの注釈が付けられています。DynamoDB Enhanced Client API が次のスニペットを使用して Person クラスのテーブルス

キーマを構築すると、API は Address および PhoneNumber クラスの使用を検出し、DynamoDB と動作する対応するマッピングを構築します。

```
TableSchema<Person> personTableSchema = TableSchema.fromBean(Person.class);
```

### ネストされたスキーマを使用する

別の方法は、次のコードに示すように、各クラスに静的テーブルスキーマビルダーを使用することです。

Address および PhoneNumber クラスのテーブルスキーマは、DynamoDB テーブルでは使用できないという意味で抽象的です。これは、プライマリキーの定義が不足しているためです。ただし、Person クラスのテーブルスキーマではネストされたスキーマとして使用されます。

PERSON\_TABLE\_SCHEMA の定義のコメント 1 行目と 2 行目の後に、抽象テーブルスキーマを使用するコードが表示されます。EnhanceType.documentOf(...) メソッド内で documentOf を使用しても、メソッドが拡張ドキュメント API の EnhancedDocument タイプを返すことを示すわけではありません。このコンテキストの documentOf(...) メソッドは、テーブルスキーマ引数を使用して DynamoDB テーブル属性との間でクラス引数をマッピングする方法を知っているオブジェクトを返します。

### 静的スキーマコード

```
// Abstract table schema that cannot be used to work with a DynamoDB table,  
// but can be used as a nested schema.  
public static final TableSchema<Address> TABLE_SCHEMA_ADDRESS =  
TableSchema.builder(Address.class)  
    .newItemSupplier(Address::new)  
    .addAttribute(String.class, a -> a.name("street")  
        .getter(Address::getStreet)  
        .setter(Address::setStreet))  
    .addAttribute(String.class, a -> a.name("city")  
        .getter(Address::getCity)  
        .setter(Address::setCity))  
    .addAttribute(String.class, a -> a.name("zipcode")  
        .getter(Address::getZipCode)  
        .setter(Address::setZipCode))  
    .addAttribute(String.class, a -> a.name("state")  
        .getter(Address::getState)  
        .setter(Address::setState))  
    .build();
```

```

// Abstract table schema that cannot be used to work with a DynamoDB table,
// but can be used as a nested schema.
public static final TableSchema<PhoneNumber> TABLE_SCHEMA_PHONENUMBER =
TableSchema.builder(PhoneNumber.class)
    .newItemSupplier(PhoneNumber::new)
    .addAttribute(String.class, a -> a.name("type")
        .getter(PhoneNumber::getType)
        .setter(PhoneNumber::setType))
    .addAttribute(String.class, a -> a.name("number")
        .getter(PhoneNumber::getNumber)
        .setter(PhoneNumber::setNumber))
    .build();

// A static table schema that can be used with a DynamoDB table.
// The table schema contains two nested schemas that are used to perform mapping
to/from DynamoDB.
public static final TableSchema<Person> PERSON_TABLE_SCHEMA =
    TableSchema.builder(Person.class)
        .newItemSupplier(Person::new)
        .addAttribute(Integer.class, a -> a.name("id")
            .getter(Person::getId)
            .setter(Person::setId)
            .addTag(StaticAttributeTags.primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("firstName")
            .getter(Person::getFirstName)
            .setter(Person::setFirstName))
        .addAttribute(String.class, a -> a.name("lastName")
            .getter(Person::getLastName)
            .setter(Person::setLastName))
        .addAttribute(Integer.class, a -> a.name("age")
            .getter(Person::getAge)
            .setter(Person::setAge))
        .addAttribute(EnhancedType.listOf(String.class), a ->
a.name("hobbies")
            .getter(Person::getHobbies)
            .setter(Person::setHobbies))
        .addAttribute(EnhancedType.mapOf(
            EnhancedType.of(String.class),
            // 1. Use mapping functionality of the Address table
schema.
            EnhancedType.documentOf(Address.class,
TABLE_SCHEMA_ADDRESS)), a -> a.name("addresses")
            .getter(Person::getAddresses)
            .setter(Person::setAddresses))

```

```

        .addAttribute(EnhancedType.listOf(
            // 2. Use mapping functionality of the PhoneNumber table
            schema.
                EnhancedType.documentOf(PhoneNumber.class,
                    TABLE_SCHEMA_PHONENUMBER)), a -> a.name("phoneNumbers")
                .getter(Person::getPhoneNumbers)
                .setter(Person::setPhoneNumbers))
        .build();

```

## プロジェクトのネストされた属性

`query()` および `scan()` メソッドでは、`addNestedAttributeToProject()` や `attributesToProject()` などのメソッドの呼び出しを使用して、結果で返される属性を指定できます。DynamoDB 拡張クライアント API は、リクエストが送信される前に Java メソッド呼び出しパラメータを [プロジェクト式](#) に変換します。

次の例では、Person テーブルに 2 つの項目を設定し、3 回のスキャンオペレーションを実行します。

1 回目のスキャンでは、結果を他のスキャンオペレーションと比較するために、テーブル内のすべての項目にアクセスします。

2 回目のスキャンでは、[addNestedAttributeToProject\(\)](#) ビルダーメソッドを使用して `street` 属性値のみを返します。

3 回目のスキャンオペレーションでは、[attributesToProject\(\)](#) ビルダーメソッドを使用して第 1 レベルの属性のデータ、`hobbies` を返します。`hobbies` の属性タイプはリストです。個々のリスト項目にアクセスするには、リストに対して `get()` オペレーションを実行します。

```

    personDynamoDbTable = getDynamoDbEnhancedClient().table("Person",
PERSON_TABLE_SCHEMA);
    PersonUtils.createPersonTable(personDynamoDbTable, getDynamoDbClient());
    // Use a utility class to add items to the Person table.
    List<Person> personList = PersonUtils.getItemsForCount(2);
    // This utility method performs a put against DynamoDB to save the instances in
the list argument.
    PersonUtils.putCollection(getDynamoDbEnhancedClient(), personList,
personDynamoDbTable);

    // The first scan logs all items in the table to compare to the results of the
subsequent scans.
    final PageIterable<Person> allItems = personDynamoDbTable.scan();

```

```

    allItems.items().forEach(p ->
        // 1. Log what is in the table.
        logger.info(p.toString()));

    // Scan for nested attributes.
    PageIterable<Person> streetScanResult = personDynamoDbTable.scan(b -> b
        // Use the 'addNestedAttributeToProject()' or
'addNestedAttributesToProject()' to access data nested in maps in DynamoDB.
        .addNestedAttributeToProject(
            NestedAttributeName.create("addresses", "work", "street")
        ));

    streetScanResult.items().forEach(p ->
        //2. Log the results of requesting nested attributes.
        logger.info(p.toString()));

    // Scan for a top-level list attribute.
    PageIterable<Person> phoneNumbersScanResult = personDynamoDbTable.scan(b -> b
        // Use the 'attributesToProject()' method to access first-level
attributes.
        .attributesToProject("hobbies"));

    phoneNumbersScanResult.items().forEach((p) -> {
        // 3. Log the results of the request for the 'hobbies' attribute.
        logger.info(p.toString());
        // To access an item in a list, first get the parent attribute, 'hobbies',
then access items in the list.
        String hobby = p.getHobbies().get(1);
        // 4. Log an item in the list.
        logger.info(hobby);
    });

```

```

// Logged results from comment line 1.
Person{id=2, firstName='first name 2', lastName='last name 2', age=11,
addresses={work=Address{street='street 21', city='city 21', state='state 21',
zipCode='33333'}, home=Address{street='street 2', city='city 2', state='state 2',
zipCode='22222'}}, phoneNumbers=[PhoneNumber{type='home', number='222-222-2222'},
PhoneNumber{type='work', number='333-333-3333'}], hobbies=[hobby 2, hobby 21]}
Person{id=1, firstName='first name 1', lastName='last name 1', age=11,
addresses={work=Address{street='street 11', city='city 11', state='state 11',
zipCode='22222'}, home=Address{street='street 1', city='city 1', state='state 1',
zipCode='11111'}}, phoneNumbers=[PhoneNumber{type='home', number='111-111-1111'},
PhoneNumber{type='work', number='222-222-2222'}], hobbies=[hobby 1, hobby 11]}

```

```
// Logged results from comment line 2.
Person{id=null, firstName='null', lastName='null', age=null,
  addresses={work=Address{street='street 21', city='null', state='null',
  zipCode='null'}}}, phoneNumbers=null, hobbies=null}
Person{id=null, firstName='null', lastName='null', age=null,
  addresses={work=Address{street='street 11', city='null', state='null',
  zipCode='null'}}}, phoneNumbers=null, hobbies=null}

// Logged results from comment lines 3 and 4.
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
  phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
hobby 21
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
  phoneNumbers=null, hobbies=[hobby 1, hobby 11]}
hobby 11
```

### Note

その `attributesToProject()` メソッドが、プロジェクションする属性を追加する他のビルダーメソッドに続く場合は、`attributesToProject()` に渡された属性名のリストは他のすべての属性名を置き換えます。

次のスニペット `ScanEnhancedRequest` のインスタンスで実行されたスキャンでは、ホビーデータのみが返されます。

```
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]}
```

以下のコード例では、最初に `attributesToProject()` メソッドを使用します。この順序付けでは、要求された他のすべての属性が保持されます。

```
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]}
```

**@DynamoDbPreserveEmptyObject** で空のオブジェクトを保存します。

空のオブジェクトを含む Bean を Amazon DynamoDB に保存し、取得時に SDK に空のオブジェクトを再作成させたい場合は、内部 Bean のゲッターに `@DynamoDbPreserveEmptyObject` の注釈を付けます。

注釈の仕組みを説明するために、コード例では次の 2 つの bean を使用しています。

例: Bean

次のデータクラスには 2 つの InnerBean フィールドがあります。getter メソッド、`getInnerBeanWithoutAnno()` には `@DynamoDbPreserveEmptyObject` という注釈が付いていません。`getInnerBeanWithAnno()` メソッドには注釈が付けられています。



```
@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();
    }
}
```

以下の InnerBean クラスのインスタンスは MyBean フィールドであり、サンプルコードで空のオブジェクトとして初期化されます。

```
@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 + '\'' +
        '}';
}
}
```

次のコード例では、初期化された内部 Bean を含む MyBean オブジェクトを DynamoDB に保存し、その項目を取得します。ログ出力には、innerBeanWithoutAnno は初期化されておらず、innerBeanWithAnno が作成されたことが示されています。

```
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;
}
```

## 代わりに静的スキーマ

Bean の注釈の代わりに、以下の `StaticTableSchema` バージョンのテーブルスキーマを使用できます。

```
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();
}
```

## ネストされたオブジェクトの NULL 属性の保存を避ける

`@DynamoDbIgnoreNulls` 注釈を適用することで、データクラスオブジェクトを DynamoDB に保存するときに、ネストされたオブジェクトの null 属性をスキップできます。これとは対照的に、NULL 値を持つ最上位の属性はデータベースに保存されません。

注釈の仕組みを説明するために、コード例では次の 2 つの Bean を使用しています。

例: Bean

次のデータクラスには 2 つの InnerBean フィールドがあります。getter メソッド、`getInnerBeanWithoutAnno()` には注釈が付けられています。getter `getInnerBeanWithIgnoreNullsAnno()` メソッドには `@DynamoDbIgnoreNulls` という注釈が付けられています。

```
@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();
    }
}
```

```
}
```

次の InnerBean クラスのインスタンスは MyBean のフィールドであり、次のコード例で使用されています。

```
@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();
    }
}
```

次のコード例では、InnerBean オブジェクトを作成し、その 2 つの属性のうち 1 つだけに値を設定します。

```
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);
    }
```

DynamoDB に送信される低レベルのデータを視覚化するために、コードは MyBean オブジェクトを保存する前に属性マップをログ記録します。

ログ記録された出力を見ると、`innerBeanWithIgnoreNullsAnno` が 1 つの属性を出力していることが分かります。

```
innerBeanWithIgnoreNullsAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200)})
```

`innerBeanWithoutAnno` インスタンスは 2 つの属性を出力します。1 つの属性の値は 200 で、もう 1 つは NULL 値の属性です。

```
innerBeanWithoutAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200),
innerBeanFieldString=AttributeValue(NUL=true)})
```

## 属性マップの JSON 表現

次の JSON 表現を使用すると、DynamoDB に保存されたデータを簡単に確認できます。

```
{
  "id": {
    "S": "1"
  },
  "innerBeanWithIgnoreNullsAnno": {
    "M": {
      "innerBeanFieldInteger": {
        "N": "200"
      }
    }
  }
}
```

```
    }
  },
  "innerBeanWithoutAnno": {
    "M": {
      "innerBeanFieldInteger": {
        "N": "200"
      },
      "innerBeanFieldString": {
        "NULL": true
      }
    }
  }
}
```

## 代わりに静的スキーマ

次の `StaticTableSchema` バージョンのテーブルスキーマをデータクラス注釈の代わりに使用できません。

```
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();
}
```

## DynamoDB 用の拡張ドキュメント API を使用して JSON ドキュメントを操作する

AWS SDK for Java 2.x 用 [拡張ドキュメント API](#) は、固定スキーマのないドキュメント指向のデータを扱うように設計されています。ただし、カスタムクラスを使用して個々の属性をマップすることもできます。

[拡張ドキュメント API](#) は AWS SDK for Java v1.x のドキュメント API の後継です。

### 目次

- [拡張ドキュメント API の使用を開始する](#)
  - [DocumentTableSchema および DynamoDbTable を作成する](#)
- [拡張ドキュメントを構築する](#)
  - [JSON 文字列から構築する](#)
  - [個々の要素から構築する](#)
- [CRUD オペレーションを実行する](#)
- [拡張ドキュメント属性にはカスタムオブジェクトとしてアクセスできます。](#)
- [DynamoDB を使用せずに EnhancedDocument を使用する](#)

### 拡張ドキュメント API の使用を開始する

拡張ドキュメント API には、DynamoDB 拡張クライアント API に必要なものと同じ [依存関係](#) が必要です。また、このトピックの冒頭で示したように、[DynamoDbEnhancedClient インスタンス](#) も必要です。

拡張ドキュメント API は AWS SDK for Java 2.x のバージョン 2.20.3 でリリースされたため、そのバージョンかそれ以降が必要です。



## DocumentTableSchema および DynamoDbTable を作成する

拡張ドキュメント API を使用して DynamoDB テーブルに対してコマンドを呼び出すには、テーブルをクライアント側の [DynamoDbTable<EnhancedDocument>](#) リソースオブジェクトに関連付けます。

拡張クライアントの `table()` メソッドは `DynamoDbTable<EnhancedDocument>` インスタンスを作成し、DynamoDB テーブル名と `DocumentTableSchema` のパラメータを必要とします。

[DocumentTableSchema](#) のビルダーには、プライマリインデックスキーと 1 つ以上の属性コンバータープロバイダーが必要です。この `AttributeConverterProvider.defaultProvider()` メソッドは [デフォルトタイプ](#) のコンバーターを提供します。カスタム属性コンバータープロバイダーを提供する場合でも、指定する必要があります。オプションのセカンダリインデックスキーをビルダーに追加できます。

次のコードスニペットは、スキーマレス `person` オブジェクトを格納する DynamoDB `EnhancedDocument` テーブルのクライアント側表現を生成するコードを示しています。

```
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.
```

このセクション全体で使用される `person` オブジェクトの JSON 表現を以下に示します。

### JSON `person` オブジェクト

```
{
  "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"
  }
]
}
```

## 拡張ドキュメントを構築する

[EnhancedDocument](#) は、ネストされた属性を持ち、複雑な構造を持つドキュメントタイプのオブジェクトを表します。EnhancedDocument には、DocumentTableSchema に指定されたプライマリキー属性と一致する最上位の属性が必要です。残りの内容は任意で、最上位の属性だけでなく、深くネストされた属性で構成することもできます。

EnhancedDocument インスタンスを作成するには、複数の方法で要素を追加できるビルダーを使用します。

## JSON 文字列から構築する

JSON 文字列を使用して、1 つのメソッド呼び出しに `EnhancedDocument` を構築できます。次のスニペットは、`jsonPerson()` ヘルパーメソッドから返された JSON 文字列から `EnhancedDocument` を作成します。この `jsonPerson()` メソッドは、前に示した [person オブジェクト](#) の JSON 文字列バージョンを返します。

```
EnhancedDocument document =
    EnhancedDocument.builder()
        .json( jsonPerson() )
        .build();
```

## 個々の要素から構築する

また、ビルダーのタイプセーフメソッドを使用して個々のコンポーネントから `EnhancedDocument` インスタンスを構築することもできます。

次の例では、前の例の JSON 文字列から作成された拡張ドキュメントと同様の `person` 拡張ドキュメントを作成します。

```
    /* 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();

```

## ヘルパーメソッド

```

private static String phoneNumbersJSONString() {
    return "[" +
        "{" +
        "    \"type\": \"Home\"," +
        "    \"number\": \"555-0140\"" +
        "}," +
        "{" +
        "    \"type\": \"Work\"," +
        "    \"number\": \"555-0155\"" +
        "}" +
        " ]";
}

private static Map<String, Map<String, String>> getAddresses() {
    return Map.of(
        "home", Map.of(
            "zipCode", "00002",
            "city", "Any Town",
            "state", "ME",
            "street", "123 Any Street"));
}

```

## CRUD オペレーションを実行する

EnhancedDocument インスタンスを定義したら、DynamoDB テーブルに保存できます。次のコードスニペットでは、個々の要素から作成された [personDocument](#) を使用しています。

```
documentDynamoDbTable.putItem(personDocument);
```

DynamoDB から拡張ドキュメントインスタンスを読み取った後、personDocument から保存されたデータにアクセスする次のコードスニペットに示すように、ゲッターを使用して個々の属性値を抽出することができます。または、サンプルコードの最後の部分に示されているように、コンテンツ全体を JSON 文字列に抽出することもできます。

```
// 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 インスタンスは、マッピングされたデータクラスの代わりに、[DynamoDbTable](#) または [DynamoDbEnhancedClient](#) の任意のメソッドで使用できます。

拡張ドキュメント属性にはカスタムオブジェクトとしてアクセスできます。

拡張ドキュメント API では、スキーマレス構造の属性を読み書きするための API を提供するだけでなく、カスタムクラスのインスタンスとの間で属性を変換できます。

拡張ドキュメント API は、DynamoDB 拡張クライアント API の一部として[コントロール属性変換](#)セクションに表示された `AttributeConverterProvider` と `AttributeConverter` を使用します。

次の例では、`CustomAttributeConverterProvider` をネストされた `AddressConverter` クラスとともに使用して `Address` オブジェクトを変換します。

この例は、クラスからのデータと、必要に応じて構築された構造からのデータを組み合わせることができることを示しています。この例はまた、カスタムクラスはネストされた構造のどのレベルでも使用できることを示しています。この例の `Address` オブジェクトはマップで使用される値です。

```
public static void attributeToAddressClassMappingExample(DynamoDbEnhancedClient
enhancedClient, DynamoDbClient standardClient) {
    String tableName = "customer";

    // Define the DynamoDbTable for an enhanced document.
    // The schema builder provides methods for attribute converter providers and
keys.
    DynamoDbTable<EnhancedDocument> documentDynamoDbTable =
enhancedClient.table(tableName,
        DocumentTableSchema.builder()
            // Add the CustomAttributeConverterProvider along with the
default when you build the table schema.
            .attributeConverterProviders(
                List.of(
                    new CustomAttributeConverterProvider(),
                    AttributeConverterProvider.defaultProvider()))
            .addIndexPartitionKey(TableMetadata.primaryIndexName(), "id",
AttributeValueType.N)
            .addIndexSortKey(TableMetadata.primaryIndexName(), "lastName",
AttributeValueType.S)
            .build());
    // Create the DynamoDB table if needed.
    documentDynamoDbTable.createTable();
    waitForTableCreation(tableName, standardClient);

    // The getAddressessForCustomMappingExample() helper method that provides
'addresses' shows the use of a custom Address class
    // rather than using a Map<String, Map<String, String> to hold the address
data.
    Map<String, Address> addresses = getAddressessForCustomMappingExample();

    // Build an EnhancedDocument instance to save an item with a mix of structures
defined as needed and static classes.
    EnhancedDocument personDocument = EnhancedDocument.builder()
        .putNumber("id", 50)
        .putString("firstName", "Shirley")
        .putString("lastName", "Rodriguez")
        .putNumber("age", 53)
        .putNull("nullAttribute")
        .putJson("phoneNumbers", phoneNumbersJSONString())
        // Note the use of 'EnhancedType.of(Address.class)' instead of the more
generic
}
```

```

        // 'EnhancedType.mapOf(EnhancedType.of(String.class),
EnhancedType.of(String.class))' that was used in a previous example.
        .putMap("addresses", addresses, EnhancedType.of(String.class),
EnhancedType.of(Address.class))
        .putList("hobbies", List.of("Hobby 1", "Hobby 2"),
EnhancedType.of(String.class))
        .build();
    // Save the item to DynamoDB.
    documentDynamoDbTable.putItem(personDocument);

    // Retrieve the item just saved.
    EnhancedDocument srPerson =
documentDynamoDbTable.getItem(Key.builder().partitionValue(50).sortValue("Rodriguez").build());

    // Access the addresses attribute.
    Map<String, Address> srAddresses = srPerson.get("addresses",
        EnhancedType.mapOf(EnhancedType.of(String.class),
EnhancedType.of(Address.class)));

    srAddresses.keySet().forEach(k -> logger.info(addresses.get(k).toString()));

    documentDynamoDbTable.deleteTable();

// The content logged to the console shows that the saved maps were converted to
Address instances.
Address{street='123 Main Street', city='Any Town', state='NC', zipCode='00000'}
Address{street='100 Any Street', city='Any Town', state='NC', zipCode='00000'}

```

## CustomAttributeConverterProvider コード

```

public class CustomAttributeConverterProvider implements AttributeConverterProvider {

    private final Map<EnhancedType<?>, AttributeConverter<?>> converterCache =
ImmutableMap.of(
        // 1. Add AddressConverter to the internal cache.
        EnhancedType.of(Address.class), new AddressConverter());

    public static CustomAttributeConverterProvider create() {
        return new CustomAttributeConverterProvider();
    }

    // 2. The enhanced client queries the provider for attribute converters if it
    // encounters a type that it does not know how to convert.

```



```
@SuppressWarnings("unchecked")
@Override
public <T> AttributeConverter<T> converterFor(EnhancedType<T> enhancedType) {
    return (AttributeConverter<T>) converterCache.get(enhancedType);
}

// 3. Custom attribute converter
private class AddressConverter implements AttributeConverter<Address> {
    // 4. Transform an Address object into a DynamoDB map.
    @Override
    public AttributeValue transformFrom(Address address) {

        Map<String, AttributeValue> attributeValueMap = Map.of(
            "street", AttributeValue.fromS(address.getStreet()),
            "city", AttributeValue.fromS(address.getCity()),
            "state", AttributeValue.fromS(address.getState()),
            "zipCode", AttributeValue.fromS(address.getZipCode()));

        return AttributeValue.fromM(attributeValueMap);
    }

    // 5. Transform the DynamoDB map attribute to an Address object.
    @Override
    public Address transformTo(AttributeValue attributeValue) {
        Map<String, AttributeValue> m = attributeValue.m();
        Address address = new Address();
        address.setStreet(m.get("street").s());
        address.setCity(m.get("city").s());
        address.setState(m.get("state").s());
        address.setZipCode(m.get("zipCode").s());

        return address;
    }

    @Override
    public EnhancedType<Address> type() {
        return EnhancedType.of(Address.class);
    }

    @Override
    public AttributeValueType attributeValueType() {
        return AttributeValueType.M;
    }
}
```

```
}
```

## Address クラス

```
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;
    }
}
```

```
}  
}
```

## アドレスを提供するヘルパーメソッド

次のヘルパーメソッドは、値にジェネリック Map<String, String> インスタンスではなく、値にカスタム Address インスタンスを使用するマップを提供します。

```
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);  
}
```

## DynamoDB を使用せずに EnhancedDocument を使用する

通常、EnhancedDocument のインスタンスはドキュメントタイプの DynamoDB アイテムの読み取りと書き込みに使用しますが、DynamoDB とは独立して使用することもできます。

JSON 文字列やカスタムオブジェクトを、次の例のように AttributeValues の低レベルのマップに変換できるようにするために EnhancedDocuments を使用できます。

```
public static void conversionWithoutDynamoDbExample() {  
    Address address = new Address();  
    address.setCity("my city");  
    address.setState("my state");  
    address.setStreet("my street");  
    address.setZipCode("00000");  
  
    // Build an EnhancedDocument instance for its conversion functionality alone.  
    EnhancedDocument addressEnhancedDoc = EnhancedDocument.builder()
```

```
// Important: You must specify attribute converter providers when you
build an EnhancedDocument instance not used with a DynamoDB table.
    .attributeConverterProviders(new CustomAttributeConverterProvider(),
DefaultAttributeConverterProvider.create())
    .put("addressDoc", address, Address.class)
    .build();

// Convert address to a low-level item representation.
final Map<String, AttributeValue> addressAsAttributeMap =
addressEnhancedDoc.getMapOfUnknownType("addressDoc");
logger.info("addressAsAttributeMap: {}", addressAsAttributeMap.toString());

// Convert address to a JSON string.
String addressAsJsonString = addressEnhancedDoc.getJson("addressDoc");
logger.info("addressAsJsonString: {}", addressAsJsonString);
// Convert addressEnhancedDoc back to an Address instance.
Address addressConverted = addressEnhancedDoc.get("addressDoc",
Address.class);
logger.info("addressConverted: {}", addressConverted.toString());
}

/* Console output:
    addressAsAttributeMap: {zipCode=AttributeValue(S=00000),
state=AttributeValue(S=my state), street=AttributeValue(S=my street),
city=AttributeValue(S=my city)}
    addressAsJsonString: {"zipCode":"00000","state":"my state","street":"my
street","city":"my city"}
    addressConverted: Address{street='my street', city='my city', state='my
state', zipCode='00000'}
*/
```

### Note

DynamoDB テーブルから独立した拡張ドキュメントを使用する場合は、必ずビルダーに属性コンバータープロバイダーを明示的に設定してください。

対照的に、拡張ドキュメントを DynamoDB テーブルで使用すると、ドキュメントテーブルスキーマがコンバータープロバイダーに提供されます。

## 拡張機能を使用する

DynamoDB 拡張クライアント API は、マッピング操作以外の機能を提供するプラグイン拡張機能をサポートしています。エクステンションには、`beforeWrite()` と `afterRead()` の 2 つのフックメソッドがあります。`beforeWrite()` は書き込み操作を発生する前に変更し、`afterRead()` メソッドは読み取り操作が行われた後にその結果を変更します。一部のオペレーション (アイテムの更新など) は書き込みと読み取りの両方を実行するため、両方のフックメソッドが呼び出されます。

拡張機能は、拡張クライアントビルダーで指定された順序でロードされます。1 つのエクステンションが以前のエクステンションによって変換された値に作用する可能性があるため、ロード順序は重要な場合があります。

拡張クライアント API には、[extensions](#) パッケージに含まれる一連のプラグイン拡張が付属しています。デフォルトでは、拡張クライアントは [VersionedRecordExtension](#) と [AtomicCounterExtension](#) をロードします。Enhance クライアントビルダーでデフォルトの動作を上書きして、任意の拡張機能を読み込むことができます。デフォルトの拡張機能を使いたくない場合は、`none` を指定することもできます。

独自の拡張機能をロードしても、拡張クライアントはデフォルトの拡張機能をロードしません。いずれかのデフォルトの拡張機能と同じ動作をさせたい場合は、拡張機能のリストに明示的に追加する必要があります。

次の例では、`verifyChecksumExtension` という名前のカスタム拡張機能が `VersionedRecordExtension` の後にロードされます。通常、カスタム拡張機能はデフォルトで単独でロードされます。この例では、`AtomicCounterExtension` はロードされていません。

```
DynamoDbEnhancedClientExtension versionedRecordExtension =
    VersionedRecordExtension.builder().build();

DynamoDbEnhancedClient enhancedClient =
    DynamoDbEnhancedClient.builder()
        .dynamoDbClient(dynamoDbClient)
        .extensions(versionedRecordExtension,
verifyChecksumExtension)
        .build();
```

### VersionedRecordExtension

`VersionedRecordExtension` はデフォルトでロードされ、アイテムがデータベースに書き込まれるとアイテムのバージョン番号が増加して追跡されます。実際に保存されているアイテムのバージョ

ン番号がアプリケーションが最後に読み取った値と一致しない場合、書き込みが失敗する原因となる条件が書き込みごとに追加されます。この動作により、アイテムの更新を楽観的ロックに効果的に適用できます。最初のプロセスが項目を読み取ってから更新を書き込むまでの間に、別のプロセスが項目を更新すると、書き込みは失敗します。

アイテムのバージョン番号を追跡するために使用する属性を指定するには、テーブルスキーマの数値属性にタグを付けます。

次のスニペットでは、`version` 属性がアイテムのバージョン番号を保持するように指定しています。

```
@DynamoDbVersionAttribute
public Integer getVersion() {...};
public void setVersion(Integer version) {...};
```

これと同等の静的テーブルスキーマのアプローチは、次のスニペットで示されます。

```
.addAttribute(Integer.class, a -> a.name("version")
    .getter(Customer::getVersion)
    .setter(Customer::setVersion)
    // Apply the 'version' tag to the attribute.

.tags(VersionedRecordExtension.AttributeTags.versionAttribute())
```

## AtomicCounterExtension

`AtomicCounterExtension` はデフォルトでロードされ、レコードがデータベースに書き込まれるたびにタグ付きの数値属性が増えます。開始値と増分値を指定できます。値を指定しない場合、開始値は 0 に設定され、属性の値は 1 ずつ増加します。

どの属性がカウンターかを指定するには、テーブルスキーマのタイプ `Long` の属性にタグを付けます。

次のスニペットは、`counter` 属性のデフォルトの開始値と増分値の使用方法を示しています。

```
@DynamoDbAtomicCounter
public Long getCounter() {...};
public void setCounter(Long counter) {...};
```

次のスニペットは、静的テーブルスキーマのアプローチを示しています。アトミックカウンタ拡張機能は、開始値を 10 に設定し、レコードが書き込まれるたびに値を 5 ずつ増やします。

```

.addAttribute(Integer.class, a -> a.name("counter")
    .getter(Customer::getCounter)
    .setter(Customer::setCounter)
    // Apply the 'atomicCounter' tag to the
attribute with start and increment values.
    .tags(StaticAttributeTags.atomicCounter(10L,
5L))

```

## AutoGeneratedTimestampRecordExtension

AutoGeneratedTimestampRecordExtension は、アイテムがデータベースに正常に書き込まれるたびに、そのタイプ [Instant](#) のタグ付き属性を現在のタイムスタンプで自動的に更新します。

このエクステンションはデフォルトではロードされません。そのため、このトピックの最初の例で示したように、拡張クライアントをビルドするときには、この拡張をカスタム拡張機能として指定する必要があります。

現在のタイムスタンプで更新する属性を指定するには、テーブルスキーマの Instant 属性にタグを付けます。

次のスニペットでは、lastUpdate 属性が拡張機能の動作の対象となります。属性は Instant タイプでなければならないという要件に注意してください。

```

@DynamoDbAutoGeneratedTimestampAttribute
public Instant getLastUpdate() {...}
public void setLastUpdate(Instant lastUpdate) {...}

```

これと同等の静的テーブルスキーマのアプローチは、次のスニペットで示されます。

```

.addAttribute(Instant.class, a -> a.name("lastUpdate")
    .getter(Customer::getLastUpdate)
    .setter(Customer::setLastUpdate)
    // Applying the 'autoGeneratedTimestamp' tag to
the attribute.
    .tags(AutoGeneratedTimestampRecordExtension.AttributeTags.autoGeneratedTimestampAttribute()))

```

## カスタム拡張機能

次のカスタム拡張クラスは、更新式を使用する beforeWrite() メソッドを示しています。2 行目のコメントの後に、データベース内の項目にまだ registrationDate 属性がない場合に

registrationDate 属性を設定するための SetAction を作成します。Customer オブジェクトが更新されるたびに、拡張機能は registrationDate が設定されていることを確認します。

```
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();
    }
}
```



## DynamoDB 拡張クライアント API を非同期的に使用する

アプリケーションで DynamoDB へのノンブロッキングの非同期呼び出しが必要な場合は、[DynamoDbEnhancedAsyncClient](#) を使用できます。同期実装と似ていますが、次の主な違いがあります。

1. `DynamoDbEnhancedAsyncClient` を構築するときは、次のスニペットに示すように、標準クライアントの非同期バージョン、`DynamoDbAsyncClient` を指定する必要があります。

```
DynamoDbEnhancedAsyncClient enhancedClient =
    DynamoDbEnhancedAsyncClient.builder()
        .dynamoDbClient(dynamoDbAsyncClient)
        .build();
```

2. 1 つのデータオブジェクトを返すメソッドは、結果だけではなく結果の `CompletableFuture` を返します。そうすることで、アプリケーションは結果をブロックせずに他の処理を行うことができます。次のスニペットは、非同期 `getItem()` メソッドを示しています。

```
CompletableFuture<Customer> result = customerDynamoDbTable.getItem(customer);
// Perform other work here.
return result.join(); // Now block and wait for the result.
```

3. ページ分割された結果リストを返すメソッドは、同期 `DynamoDbEnhanceClient` が同じメソッドで返す `SdkIterable` の代わりに、`SdkPublisher` を返します。その後、アプリケーションはハンドラーをそのパブリッシャーにサブスクライブして、結果をブロックせずに非同期的に処理できます。

```
PagePublisher<Customer> results = customerDynamoDbTable.query(r ->
    r.queryConditional(keyEqualTo(k -> k.partitionValue("Smith"))));
results.subscribe(myCustomerResultsProcessor);
// Perform other work and let the processor handle the results asynchronously.
```

`SdkPublisher` API のより詳細な使用例については、このガイドの非同期 `scan()` メソッドについて説明しているセクションの[例](#)を参照してください。

## データクラス注釈

次の表は、データクラスで利用できる注釈の一覧と、このガイドの情報と例へのリンクを示しています。テーブルは、注釈名別にアルファベット順に昇順に並べ替えられます。

## このガイドで使用されているデータクラス注釈

注釈名	注釈は <sup>1</sup> に適用されます。	その内容	このガイドに記載されている場所
DynamoDbAtomicCounter	属性 <sup>2</sup>	レコードがデータベースに書き込まれるたびに、タグ付きの数値属性を増やします。	<a href="#">概要とディスカッション</a> 。
DynamoDbAttribute	属性	DynamoDB テーブル属性にマップされる Bean プロパティを定義または名前を変更します。	<ul style="list-style-type: none"> <li>• <a href="#">最初のディスカッション</a>。</li> <li>• <a href="#">「はじめに」セクション — 注釈を参照してください</a>。</li> <li>• <a href="#">クラス MovieActor In Query メソッドの例</a>。</li> </ul>
DynamoDbAutoGeneratedTimestampAttribute	属性	項目がデータベースに正常に書き込まれるたびに、タグ付き属性を現在のタイムスタンプで更新します。	<a href="#">概要とディスカッション</a> 。
DynamoDbBean	class	データクラスをテーブルスキーマにマッピング可能としてマークします。	「はじめに」セクションの <a href="#">カスタムクラス</a> で初めに使用します。このガイドでは、いくつかの使用法が説明されています。
DynamoDbConvertedBy	属性	カスタム Attribute Converter を注釈	<a href="#">最初のディスカッションと例</a> 。

注釈名	注釈は <sup>1</sup> に適用されます。	その内容	このガイドに記載されている場所
		付きの属性に関連付けます。	
DynamoDbFlatten	属性	個別の DynamoDB データクラスのすべての属性をフラット化し、データベースに読み書きされるレコードに最上位の属性として追加します。	<ul style="list-style-type: none"> <li>• <a href="#">最初のディスカッション</a>。</li> <li>• <a href="#">他のコードへの影響</a>。</li> </ul>
DynamoDbIgnore	属性	その結果、属性はマップされないままの状態となります。	<ul style="list-style-type: none"> <li>• <a href="#">最初のディスカッション</a>。</li> <li>• <a href="#">クラスで ProductCatalog を使用します</a>。</li> </ul>
DynamoDbIgnoreNulls	属性	ネストされた DynamoDb オブジェクトの NULL 属性の保存を禁止します。	<a href="#">ディスカッションと例</a> 。
DynamoDbImmutable	class	変更不可能なデータクラスをテーブルスキーマにマッピング可能としてマークします。	<ul style="list-style-type: none"> <li>• <a href="#">注釈の概要</a>。</li> <li>• <a href="#">クラスで ProductCatalog を使用します</a>。</li> <li>• <a href="#">Lombok で使用します</a>。</li> </ul>

注釈名	注釈は <sup>1</sup> に適用されます。	その内容	このガイドに記載されている場所
DynamoDbPartitionKey	属性	属性をテーブルのプライマリパーティションキー (ハッシュキー) としてマークします DynamoDb。	<ul style="list-style-type: none"> <li>• <a href="#">「はじめに」セクションのカスタマークラスでの最初の使用。</a></li> <li>• <a href="#">Lombok の使用。</a></li> </ul>
DynamoDbP reserveEmptyObject	属性	注釈付き属性にマップされたオブジェクトにデータが存在しない場合、オブジェクトはすべて NULL フィールドで初期化されるように指定します。	<a href="#">ディスカッションと例。</a>
DynamoDbS econdaryPartitionKey	属性	グローバルセカンダリインデックスのパーティションキーとして属性をマークします。	<ul style="list-style-type: none"> <li>• <a href="#">セカンダリインデックスと例で使用します。</a></li> <li>• <a href="#">クエリメソッドの例。</a></li> <li>• <a href="#">Lombok の例</a></li> <li>• <a href="#">変更不可能クラスの使用。</a></li> </ul>
DynamoDbS econdarySortKey	属性	属性をグローバルまたはローカルのセカンダリインデックスのオプションのソートキーとしてマークします。	<ul style="list-style-type: none"> <li>• <a href="#">セカンダリインデックスと例で使用します。</a></li> <li>• <a href="#">クエリメソッドの例。</a></li> <li>• <a href="#">Lombok の例。</a></li> <li>• <a href="#">変更不可能クラスの使用。</a></li> </ul>

注釈名	注釈は <sup>1</sup> に適用されます。	その内容	このガイドに記載されている場所
DynamoDbSortKey	属性	属性をオプションのプライマリソートキー (レンジキー) としてマークします。	<ul style="list-style-type: none"> <li>• <a href="#">カスタマークラスの「はじめに」セクション。</a></li> <li>• <a href="#">変更不可能クラスの使用。</a></li> <li>• <a href="#">Lombok の例。</a></li> <li>• <a href="#">クエリメソッドの例。</a></li> </ul>
DynamoDbUpdateBehavior	属性	などの「update」オペレーションの一部としてこの属性が更新されるとき、動作を指定します UpdateItem。	<a href="#">概要と例。</a>
DynamoDbVersionAttribute	属性	アイテムのバージョン番号を増やします。	<a href="#">概要とディスカッション。</a>

<sup>1</sup>属性レベルの注釈はゲッターまたはセッターに適用できますが、両方には適用できません。このガイドでは、ゲッターの注釈について説明します。

<sup>2</sup>用語は通常property、JavaBean データクラスにカプセル化された値に使用されます。ただし、このガイドでは、DynamoDB で使用されている用語との一貫性を保つため、代わりに attribute という用語を使用しています。

## の操作 Amazon EC2

このセクションでは、AWS SDK for Java 2.x [Amazon EC2](#)を使用するプログラミングの例を示します。

### トピック

- [Amazon EC2 インスタンスを管理する](#)

- [AWS リージョン とアベイラビリティゾーンの使用](#)
- [でセキュリティグループを操作する Amazon EC2](#)
- [Amazon EC2 インスタンスメタデータを操作する](#)

## Amazon EC2 インスタンスを管理する

### インスタンスを作成する

新しい Amazon EC2 インスタンスを作成するには、[Ec2Client](#) の [runInstances](#) メソッドを呼び出して、使用する [Amazon マシンイメージ \(AMI\)](#)、および [インスタンスタイプ](#) を含む [RunInstancesRequest](#) を指定します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

### Code

```
public static String createEC2Instance(Ec2Client ec2, String name, String amiId ) {

    RunInstancesRequest runRequest = RunInstancesRequest.builder()
        .imageId(amiId)
        .instanceType(InstanceType.T1_MICRO)
        .maxCount(1)
        .minCount(1)
        .build();

    RunInstancesResponse response = ec2.runInstances(runRequest);
    String instanceId = response.instances().get(0).instanceId();

    Tag tag = Tag.builder()
        .key("Name")
        .value(name)
        .build();
```

```
    CreateTagsRequest tagRequest = CreateTagsRequest.builder()
        .resources(instanceId)
        .tags(tag)
        .build();

    try {
        ec2.createTags(tagRequest);
        System.out.printf(
            "Successfully started EC2 Instance %s based on AMI %s",
            instanceId, amiId);

        return instanceId;
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

[GitHub](#) で完全な例をご覧ください。

## インスタンスの起動

Amazon EC2 インスタンスを起動するには、Ec2Client の [startInstances](#) メソッドを呼び出して、開始するインスタンスの ID を含む [StartInstancesRequest](#) を指定します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
```

### Code

```
public static void startInstance(Ec2Client ec2, String instanceId) {

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();
```

```
    ec2.startInstances(request);
    System.out.printf("Successfully started instance %s", instanceId);
}
```

[GitHub](#) で完全な例をご覧ください。

## インスタンスの停止

Amazon EC2 インスタンスを停止するには、Ec2Client の [stopInstances](#) メソッドを呼び出して、停止するインスタンスの ID を含む [StopInstancesRequest](#) を指定します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
```

### Code

```
public static void stopInstance(Ec2Client ec2, String instanceId) {

    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    ec2.stopInstances(request);
    System.out.printf("Successfully stopped instance %s", instanceId);
}
```

[GitHub](#) で完全な例をご覧ください。

## インスタンスの再起動

Amazon EC2 インスタンスを再起動するには、Ec2Client の [rebootInstances](#) メソッドを呼び出して、再起動するインスタンスの ID を含む [RebootInstancesRequest](#) を指定します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
```



```
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.RebootInstancesRequest;
```

## Code

```
public static void rebootEC2Instance(Ec2Client ec2, String instanceId) {

    try {
        RebootInstancesRequest request = RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        ec2.rebootInstances(request);
        System.out.printf(
            "Successfully rebooted instance %s", instanceId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

[GitHub](#) で完全な例をご覧ください。

## インスタンスの説明

インスタンスをリスト表示するには、[DescribeInstancesRequest](#) を作成し、Ec2Client の [describeInstances](#) メソッドを呼び出します。これにより [DescribeInstancesResponse](#) オブジェクトが返されます。このオブジェクトを使用して、お客様のアカウントとリージョンの Amazon EC2 インスタンスを一覧表示することができます。

インスタンスは予約ごとにグループ化されています。それぞれの予約は、インスタンスを起動した `startInstances` の呼び出しに対応しています。インスタンスを一覧表示するには、まず `DescribeInstancesResponse` クラスの `reservations` メソッドを呼び出してから、返るそれぞれの `instancesReservation` [オブジェクトの](#) を呼び出します。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Instance;
```

```
import software.amazon.awssdk.services.ec2.model.Reservation;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

## Code

```
public static void describeEC2Instances( Ec2Client ec2){

    String nextToken = null;

    try {

        do {

            DescribeInstancesRequest request =
DescribeInstancesRequest.builder().maxResults(6).nextToken(nextToken).build();
            DescribeInstancesResponse response = ec2.describeInstances(request);

            for (Reservation reservation : response.reservations()) {
                for (Instance instance : reservation.instances()) {
                    System.out.println("Instance Id is " + instance.instanceId());
                    System.out.println("Image id is "+ instance.imageId());
                    System.out.println("Instance type is "+
instance.instanceType());
                    System.out.println("Instance state name is "+
instance.state().name());
                    System.out.println("monitoring information is "+
instance.monitoring().state());

                }
            }
            nextToken = response.nextToken();
        } while (nextToken != null);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

結果はページ分割されます。さらに結果を取得するには、結果オブジェクトの `nextToken` メソッドから返る値を新しいリクエストオブジェクトの `nextToken` メソッドに渡した後、次の `describeInstances` の呼び出しでその新しいリクエストオブジェクトを使用します。

[GitHub](#) で完全な例をご覧ください。

## インスタンスのモニタリング

CPU やネットワークの使用率、使用可能なメモリ、ディスクの残り容量など、Amazon EC2 インスタンスのさまざまな側面を監視できます。インスタンスのモニタリングの詳細については、「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「[Monitoring Amazon EC2](#)」を参照してください。

インスタンスのモニタリングを開始するには、モニタリングするインスタンスの ID で [MonitorInstancesRequest](#) を作成し、Ec2Client の [monitorInstances](#) メソッドに渡します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.MonitorInstancesRequest;
import software.amazon.awssdk.services.ec2.model.UnmonitorInstancesRequest;
```

### Code

```
public static void monitorInstance( Ec2Client ec2, String instanceId) {

    MonitorInstancesRequest request = MonitorInstancesRequest.builder()
        .instanceIds(instanceId).build();

    ec2.monitorInstances(request);
    System.out.printf(
        "Successfully enabled monitoring for instance %s",
        instanceId);
}
```

[GitHub](#) で完全な例をご覧ください。

## インスタンスのモニタリング停止

インスタンスのモニタリングを停止するには、モニタリングを停止するインスタンスの ID で [UnmonitorInstancesRequest](#) を作成し、Ec2Client の [unmonitorInstances](#) メソッドに渡します。

### インポート

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.MonitorInstancesRequest;
import software.amazon.awssdk.services.ec2.model.UnmonitorInstancesRequest;
```

## Code

```
public static void unmonitorInstance(Ec2Client ec2, String instanceId) {
    UnmonitorInstancesRequest request = UnmonitorInstancesRequest.builder()
        .instanceIds(instanceId).build();

    ec2.unmonitorInstances(request);

    System.out.printf(
        "Successfully disabled monitoring for instance %s",
        instanceId);
}
```

[GitHub](#) で完全な例をご覧ください。

## 詳細情報

- Amazon EC2 API リファレンスの [RunInstances](#)
- Amazon EC2 API リファレンスの [DescribeInstances](#)
- Amazon EC2 API リファレンスの [StartInstances](#)
- Amazon EC2 API リファレンスの [StopInstances](#)
- Amazon EC2 API リファレンスの [RebootInstances](#)
- Amazon EC2 API リファレンスの [MonitorInstances](#)
- Amazon EC2 API リファレンスの [UnmonitorInstances](#)

## AWS リージョン とアベイラビリティゾーンの使用

### リージョンの記述

アカウントに使用可能なリージョンを一覧表示するには、Ec2Client の describeRegions メソッドを呼び出します。このメソッドからは、[DescribeRegionsResponse](#) が返されます。返されたオブジェクトの regions メソッドを呼び出して、各リージョンを表す [Region](#) オブジェクトの一覧を取得します。

## インポート

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

## Code

```
try {
    DescribeRegionsResponse regionsResponse = ec2.describeRegions();
    regionsResponse.regions().forEach(region -> {
        System.out.printf(
            "Found Region %s with endpoint %s%n",
            region.regionName(),
            region.endpoint());
        System.out.println();
    });
}
```

「」で[完全な例](#)を参照してください [GitHub](#)。

## アベイラビリティゾーンの詳細を表示する

アカウントに使用可能な各アベイラビリティゾーンを一覧表示するには、Ec2Client の describeAvailabilityZones メソッドを呼び出します。このメソッドからは、[DescribeAvailabilityZonesResponse](#) が返されます。availabilityZones メソッドを呼び出して、各アベイラビリティゾーンを表す [AvailabilityZone](#) オブジェクトのリストを取得します。

## インポート

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

## Code

Ec2Client を作成します。

```
software.amazon.awssdk.regions.Region region =
software.amazon.awssdk.regions.Region.US_EAST_1;
Ec2Client ec2 = Ec2Client.builder()
    .region(region)
    .build();
```

次に、describeAvailabilityZones() を呼び出して結果を取得します。

```
DescribeAvailabilityZonesResponse zonesResponse =
ec2.describeAvailabilityZones();
zonesResponse.availabilityZones().forEach(zone -> {
    System.out.printf(
        "Found Availability Zone %s with status %s in region %s%n",
        zone.zoneName(),
        zone.state(),
        zone.regionName()
    );
    System.out.println();
});
```

「」で[完全な例](#)を参照してください GitHub。

## アカウントの説明

アカウントの EC2 関連情報を一覧表示するには、Ec2Client の describeAccountAttributes メソッドを呼び出します。このメソッドは [DescribeAccountAttributesResponse](#) オブジェクトを返します。このオブジェクト accountAttributes メソッドを呼び出して、[AccountAttribute](#) オブジェクトのリストを取得します。リストを繰り返して AccountAttribute オブジェクトを取得できます。

AccountAttribute オブジェクトの attributeValues メソッドを呼び出すことで、アカウントの属性値を取得できます。このメソッドは [AccountAttributeValue](#) オブジェクトのリストを返します。この 2 番目のリストを反復処理して、属性の値を表示できます (次のコード例を参照)。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

## Code

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAccount {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        describeEC2Account(ec2);
        System.out.print("Done");
        ec2.close();
    }

    public static void describeEC2Account(Ec2Client ec2) {
        try {
            DescribeAccountAttributesResponse accountResults =
ec2.describeAccountAttributes();
            accountResults.accountAttributes().forEach(attribute -> {
                System.out.print("\n The name of the attribute is " +
attribute.attributeName());
                attribute.attributeValues().forEach(
                    myValue -> System.out.print("\n The value of the attribute is "
+ myValue.attributeValue()));
            });

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
}
```

「」で[完全な例](#)を参照してください GitHub。

## 詳細情報

- Linux インスタンス用 Amazon EC2 ユーザーガイドの[リージョンとアベイラビリティゾーン](#)
- [DescribeRegions](#) Amazon EC2 API リファレンスの
- [DescribeAvailabilityZones](#) Amazon EC2 API リファレンスの

## でセキュリティグループを操作する Amazon EC2

### セキュリティグループの作成

セキュリティグループを作成するには、キーの名前[CreateSecurityGroupRequest](#)を含む `createSecurityGroupRequest` を使用して `Ec2Client` の `createSecurityGroup` メソッドを呼び出します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.IpRange;
```

### Code

```
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
    .groupName(groupName)
    .description(groupDesc)
    .vpcId(vpcId)
    .build();

        CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);
```



「」で[完全な例](#)を参照してください GitHub。

## セキュリティグループを設定する

セキュリティグループは、Amazon EC2 インスタンスへのインバウンド (入力) トラフィックとアウトバウンド (出力) トラフィックの両方を制御できます。

セキュリティグループに進入ルールを追加するには、Ec2Client の `authorizeSecurityGroupIngress` メソッドを使用して、セキュリティグループの名前と、[AuthorizeSecurityGroupIngressRequest](#) オブジェクト内で割り当てるアクセスルール ([IpPermission](#)) を指定します。以下の例では、セキュリティグループへの IP のアクセス許可の追加方法を説明します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.IpRange;
```

### Code

最初に、Ec2Client を作成します

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

その後、Ec2Client の `authorizeSecurityGroupIngress` メソッドを使用します。

```
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 "";
}
```

セキュリティグループに Egress ルールを追加するには、`Ec2Client` の `authorizeSecurityGroupEgress` メソッド [AuthorizeSecurityGroupEgressRequest](#) に同様のデータを指定します。

「」で [完全な例](#) を参照してください GitHub。

## セキュリティグループを記述する

セキュリティグループについて記述、またはそれらに関する情報を収集するには、`Ec2Client` の `describeSecurityGroups` メソッドを呼び出します。 [DescribeSecurityGroupsResponse](#) メソッド

ドを呼び出してセキュリティグループのリストにアクセスするために使用できる を返します。これによりsecurityGroups、 [SecurityGroup](#) オブジェクトのリストが返されます。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.SecurityGroup;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

## Code

```
public static void describeEC2SecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
            DescribeSecurityGroupsRequest.builder()
                .groupIds(groupId).build();

        DescribeSecurityGroupsResponse response =
            ec2.describeSecurityGroups(request);

        for(SecurityGroup group : response.securityGroups()) {
            System.out.printf(
                "Found Security Group with id %s, " +
                "vpc id %s " +
                "and description %s",
                group.groupId(),
                group.vpcId(),
                group.description());
        }
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

「」で [完全な例](#) を参照してください [GitHub](#)。

## セキュリティグループを削除する

セキュリティグループを削除するには、Ec2Client の deleteSecurityGroup メソッドを呼び出し、削除するセキュリティグループの ID [DeleteSecurityGroupRequest](#) を含む を渡します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

### Code

```
public static void deleteEC2SecGroup(Ec2Client ec2,String groupId) {

    try {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        ec2.deleteSecurityGroup(request);
        System.out.printf(
            "Successfully deleted Security Group with id %s", groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

「」で [完全な例](#) を参照してください GitHub。

### 詳細情報

- 「Linux インスタンス用 Amazon EC2 ユーザーガイド」の [Amazon EC2 「セキュリティグループ」](#)
- 「Linux インスタンス用ユーザーガイド」の「Linux インスタンスのインバウンドトラフィックの承認 Amazon EC2 」
- [CreateSecurityGroup](#) Amazon EC2 API リファレンスの
- [DescribeSecurityGroups](#) Amazon EC2 API リファレンスの

- [DeleteSecurityGroup](#) Amazon EC2 API リファレンスの
- [AuthorizeSecurityGroupIngress](#) Amazon EC2 API リファレンスの

## Amazon EC2 インスタンスメタデータを操作する

Amazon EC2 インスタンスメタデータサービスの Java SDK クライアント (メタデータクライアント) を使用すると、アプリケーションはローカル EC2 インスタンスのメタデータにアクセスできます。メタデータクライアントは [IMDSv2](#) のローカルインスタンス (インスタンスメタデータサービス v2) と連携し、セッション指向のリクエストを使用します。

SDK には 2 つのクライアントクラスがあります。同期 [Ec2MetadataClient](#) はブロッキングオペレーション用で、[Ec2MetadataAsyncClient](#) は非同期、ノンブロッキングのユースケース用です。

### 開始方法

メタデータクライアントを使用するには、imds Maven アーティファクトをプロジェクトに追加します。クラスパスには [SdkHttpClient](#) (または非同期バリエーションの場合は [SdkAsyncHttpClient](#)) のクラスも必要です。

次の Maven XML は、同期 [URLConnectionHttpClient](#) を使用するための依存関係のスニペットと、メタデータクライアントの依存関係を示しています。

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

<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>imds</artifactId>
  </dependency>
  <dependency>
```

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

[Maven central リポジトリ](#)で bom アーティファクトの最新バージョンを検索してください。

非同期 HTTP クライアントを使用するには、`url-connection-client` アーティファクトの依存関係スニペットを置き換えてください。たとえば、次のスニペットでは [NettyNioAsyncHttpClient](#) の実装が取り込まれています。

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

## メタデータクライアントを使用する

### メタデータクライアントをインスタンス化する

クラスパスに `SdkHttpClient` インタフェースの実装が 1 つしかない場合は、同期 `Ec2MetadataClient` のインスタンスをインスタンス化できます。そのためには、以下のスニペットに示すように静的 `Ec2MetadataClient#create()` メソッドを呼び出します。

```
Ec2MetadataClient client = Ec2MetadataClient.create(); //
'Ec2MetadataAsyncClient#create' is the asynchronous version.
```

アプリケーションに `SdkHttpClient` または `SdkHttpAsyncClient` インタフェースの実装が複数ある場合は、[the section called “設定可能な HTTP クライアント”](#) セクションに示されているように、メタデータクライアントが使用する実装を指定する必要があります。

### Note

Amazon S3 などのほとんどのサービスクライアントでは、SDK for Java は `SdkHttpClient` または `SdkHttpAsyncClient` インタフェースの実装を自動的に追加します。メタデータクライアントが同じ実装を使用している場合は、`Ec2MetadataClient#create()` は問題なく動作します。別の実装が必要な場合は、メタデータクライアントの作成時にその実装を指定する必要があります。

## リクエストを送信する

インスタンスメタデータを取得するには、`Ec2MetadataClient` クラスをインスタンス化し、[インスタンスメタデータカテゴリ](#)を指定するパスパラメータを使用して `get` メソッドを呼び出します。

次の例では、`ami-id` キーに関連付けられた値をコンソールに出力します。

```
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.
```

パスが無効な場合、`get` メソッドは例外を投げます。

`close` 同じクライアントインスタンスを複数のリクエストに再利用し、リソースを解放する必要がなくなったらクライアントを呼び出します。クローズメソッドが呼び出されると、クライアントインスタンスは使用できなくなります。

## レスポンスを解析する

EC2 インスタンスメタデータはさまざまな形式で出力できます。プレーンテキストと JSON が最も一般的に使用される形式です。メタデータクライアントには、これらの形式を処理する方法が用意されています。

次の例に示すように、`asString` メソッドを使用してデータを Java 文字列として取得します。`asList` メソッドを使用して、複数行を返すプレーンテキストレスポンスを区切ることもできます。

```
Ec2MetadataClient client = Ec2MetadataClient.create();
Ec2MetadataResponse response = client.get("/latest/meta-data/");
String fullResponse = response.asString();
List<String> splits = response.asList();
```

レスポンスが JSON の場合は、次のコードスニペットに示すように、`Ec2MetadataResponse#asDocument` メソッドを使用して JSON レスポンスを解析して [ドキュメント](#) インスタンスにします。

```
Document fullResponse = response.asDocument();
```

メタデータの形式が JSON でない場合は例外が発生します。レスポンスが正常に解析されたら、[ドキュメント API](#) を使用してレスポンスをより詳細に調べることができます。JSON 形式のレスポンス

スを返すメタデータカテゴリについては、インスタンスの「[metadata category chart](#)」を参照してください。

## メタデータクライアントを設定する

### 再試行

メタデータクライアントには再試行メカニズムを設定できます。そうすることで、クライアントは予期しない理由で失敗したリクエストを自動的に再試行できます。デフォルトでは、クライアントは失敗したリクエストに対し、エクスポネンシャルバックオフ時間を置いて 3 回再試行します。

ユースケースで別の再試行メカニズムが必要な場合は、ビルダーの `retryPolicy` メソッドを使用してクライアントをカスタマイズできます。たとえば、次の例は、試行間隔が 2 秒、再試行が 5 回になるように設定された同期クライアントを示しています。

```
BackoffStrategy fixedBackoffStrategy =
    FixedDelayBackoffStrategy.create(Duration.ofSeconds(2));
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .retryPolicy(retryPolicyBuilder ->
            retryPolicyBuilder.numRetries(5)

            .backoffStrategy(fixedBackoffStrategy))
        .build();
```

メタデータクライアントで使用できる [BackoffStrategies](#) はいくつかあります。

次のスニペットに示すように、リトライメカニズムを完全に無効にすることもできます。

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .retryPolicy(Ec2MetadataRetryPolicy.none())
        .build();
```

`Ec2MetadataRetryPolicy#none()` を使用すると、メタデータクライアントが再試行しないように、デフォルトの再試行ポリシーが無効になります。

### IP バージョン

デフォルトでは、メタデータクライアントは `http://169.254.169.254` の IPv4 エンドポイントを使用します。IPv6 バージョンを使用するようにクライアントを変更するには、ビルダーの



endpointMode または endpoint メソッドを使用します。ビルダーで両方のメソッドが呼び出されると、例外が発生します。

次の例は、両方の IPV6 オプションを示しています。

```
Ec2MetadataClient client =  
    Ec2MetadataClient.builder()  
        .endpointMode(EndpointMode.IPV6)  
        .build();
```

```
Ec2MetadataClient client =  
    Ec2MetadataClient.builder()  
        .endpoint(URI.create("http://[fd00:ec2::254]"))  
        .build();
```

## 主な特徴

### 非同期クライアント

ノンブロッキングバージョンのクライアントを使用するには、Ec2MetadataAsyncClient クラスのインスタンスをインスタンス化します。次の例のコードは、デフォルト設定で非同期クライアントを作成し、get メソッドを使用して ami-id キーの値を取得します。

```
Ec2MetadataAsyncClient asyncClient = Ec2MetadataAsyncClient.create();  
CompletableFuture<Ec2MetadataResponse> response = asyncClient.get("/latest/meta-data/  
ami-id");
```

get メソッドから返された java.util.concurrent.CompletableFuture は、レスポンスが返された時点で完了します。次の例では、ami-id メタデータをコンソールに出力します。

```
response.thenAccept(metadata -> System.out.println(metadata.asString()));
```

### 設定可能な HTTP クライアント

各メタデータクライアントのビルダーには、カスタマイズされた HTTP クライアントを提供するために使用できる httpClient メソッドがあります。

次の例はカスタム HttpURLConnectionHttpClient インスタンスのコードを示しています。

```
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.
```

次の例は、非同期メタデータクライアントを使用するカスタム `NettyNioAsyncHttpClient` インスタンスのコードを示しています。

```
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.
```

このガイドの [the section called “HTTP クライアント”](#) トピックでは、SDK for Java で使用可能な HTTP クライアントの設定方法について詳しく説明しています。

## トークンキャッシュ

メタデータクライアントは IMDSv2 を使用するため、すべてのリクエストはセッションに関連付けられます。セッションは、メタデータクライアントが管理する有効期限のあるトークンによって定義されます。メタデータをリクエストするたびに、トークンは有効期限が切れるまで自動的に再利用されます。

デフォルトでは、トークンは 6 時間 (21,600 秒) 持続します。特定のユースケースで高度な設定が必要でない限り、有効期間をデフォルト値のままにしておくことをお勧めします。

必要に応じて、`tokenTtl` ビルダーメソッドを使用して期間を設定します。たとえば、次のスニペットのコードでは、セッション期間が 5 分のクライアントを作成します。

```
Ec2MetadataClient client =  
    Ec2MetadataClient.builder()  
        .tokenTtl(Duration.ofMinutes(5))  
        .build();
```

ビルダーで `tokenTtl` メソッドを呼び出さない場合は、代わりにデフォルトの継続時間である 21,600 が使用されます。

## IAM を使用する

このセクションでは、AWS SDK for Java 2.x を使用して AWS Identity and Access Management (IAM) をプログラムする例を示します。

AWS Identity and Access Management (IAM) を使用すると、AWS のサービスおよびリソースに対するお客様のユーザーのアクセスを安全にコントロールすることができます。IAM を使用すると、AWS のユーザーとグループを作成および管理し、許可を使用して AWS リソースへのアクセスを許可および拒否できます。IAM の詳細なガイドについては、「[IAM ユーザーガイド](#)」を参照してください。

次の例には各手法を示すのに必要なコードのみが含まれます。[完全なサンプルコードは GitHub で入手できます](#)。そこから、単一のソースファイルをダウンロードするカリポジトリをローカルにクローン作成して、ビルドし実行するためのすべての例を取得できます。

### トピック

- [IAM アクセスキーの管理](#)
- [IAM ユーザーを管理する](#)
- [を使用して IAM ポリシーを作成する AWS SDK for Java 2.x](#)
- [IAM ポリシーの使用](#)
- [IAM サーバー証明書の使用](#)

## IAM アクセスキーの管理

### アクセスキーの作成

IAM アクセスキーを作成するには、[CreateAccessKeyRequest](#) オブジェクトを使用して `IamClient#createAccessKey` メソッドを呼び出します。

**Note**

がグローバルサービスであるため、IamClient呼び出しが機能するには、リージョンをAWS\_GLOBAL IAM に設定する必要があります。

## インポート

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

## Code

```
public static String createIAMAccessKey(IamClient iam,String user) {

    try {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(user).build();

        CreateAccessKeyResponse response = iam.createAccessKey(request);
        String keyId = response.accessKey().accessKeyId();
        return keyId;

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

「」で[完全な例](#)を参照してください GitHub。

## アクセスキーの一覧表示

特定のユーザーのアクセスキーを一覧表示するには、キーを一覧表示するユーザー名を含む[ListAccessKeysRequest](#)オブジェクトを作成し、それを IamClient's listAccessKeys メソッドに渡します。

**Note**

にユーザー名を指定しない場合 `listAccessKeys`、リクエストに署名 AWS アカウントしたに関連付けられたアクセスキーを一覧表示しようとします。

## インポート

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

## Code

```
public static void listKeys( IamClient iam,String userName ){

    try {
        boolean done = false;
        String newMarker = null;

        while (!done) {
            ListAccessKeysResponse response;

            if(newMarker == null) {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName).build();
                response = iam.listAccessKeys(request);
            } else {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .marker(newMarker).build();
                response = iam.listAccessKeys(request);
            }

            for (AccessKeyMetadata metadata :
                response.accessKeyMetadata()) {
                System.out.format("Retrieved access key %s",
                    metadata.accessKeyId());
            }
        }
    }
}
```

```
        if (!response.isTruncated()) {
            done = true;
        } else {
            newMarker = response.marker();
        }
    }

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

`listAccessKeys` の結果はページ分割されます (デフォルトで最大 1 回の呼び出しごとに 100 レコード)。返された [ListAccessKeysResponse](#) オブジェクト `isTruncated` を呼び出して、クエリが返した結果が少なくなつてから利用可能かどうかを確認できます。利用可能な場合は、`marker` で `ListAccessKeysResponse` を呼び出して、新しいリクエストを作成するときに使用します。この新しいリクエストを `listAccessKeys` の次の呼び出しで使用します。

「」で [完全な例](#) を参照してください GitHub。

## アクセスキーの最終使用時刻の取得

アクセスキーが最後に使用された時刻を取得するには、アクセスキー `IamClient` の `getAccessKeyLastUsed` の ID ( [GetAccessKeyLastUsedRequest](#) オブジェクト) を使用して渡すことができます。

その後、返された [GetAccessKeyLastUsedResponse](#) オブジェクトを使用して、キーの最終使用時刻を取得できます。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetAccessKeyLastUsedRequest;
import software.amazon.awssdk.services.iam.model.GetAccessKeyLastUsedResponse;
import software.amazon.awssdk.services.iam.model.IamException;
```

### Code

```
public static void getAccessKeyLastUsed(IamClient iam, String accessId ){
```

```
try {
    GetAccessKeyLastUsedRequest request = GetAccessKeyLastUsedRequest.builder()
        .accessKeyId(accessId).build();

    GetAccessKeyLastUsedResponse response = iam.getAccessKeyLastUsed(request);

    System.out.println("Access key was last used at: " +
        response.accessKeyLastUsed().lastUsedDate());

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

「」で[完全な例](#)を参照してください GitHub。

## アクセスキーのアクティブ化または非アクティブ化

[UpdateAccessKeyRequest](#) オブジェクトを作成し、アクセスキー ID、オプションでユーザー名、および目的の を指定して [status](#)、リクエストオブジェクトを `IamClient.updateAccessKeyメソッド` に渡すことで、アクセスキーをアクティブ化または非アクティブ化できます。

### インポート

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

### Code

```
public static void updateKey(IamClient iam, String username, String accessId,
String status ) {

    try {
        if (status.toLowerCase().equalsIgnoreCase("active")) {
            statusType = StatusType.ACTIVE;
        } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
```

```
        statusType = StatusType.INACTIVE;
    } else {
        statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
    }
    UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
        .accessKeyId(accessId)
        .userName(username)
        .status(statusType)
        .build();

    iam.updateAccessKey(request);

    System.out.printf(
        "Successfully updated the status of access key %s to" +
        "status %s for user %s", accessId, status, username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

「」で[完全な例](#)を参照してください GitHub。

## アクセスキーの削除

アクセスキーを完全に削除するには、`IamClient`の`deleteKey`メソッドを呼び出し、アクセスキーの ID とユーザー名 [DeleteAccessKeyRequest](#) を含む を指定します。

### Note

削除してしまうと、キーは取得することも使用することもできなくなります。キーを一時的に非アクティブ化して後で再度アクティブ化できるようにするには、代わりに [updateAccessKey](#) メソッドを使用します。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
```



```
import software.amazon.awssdk.services.iam.model.IamException;
```

## Code

```
public static void deleteKey(IamClient iam ,String username, String accessKey ) {  
  
    try {  
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()  
            .accessKeyId(accessKey)  
            .userName(username)  
            .build();  
  
        iam.deleteAccessKey(request);  
        System.out.println("Successfully deleted access key " + accessKey +  
            " from user " + username);  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

「」で[完全な例](#)を参照してください GitHub。

## 詳細情報

- [CreateAccessKey](#) IAM API リファレンスの
- [ListAccessKeys](#) IAM API リファレンスの
- [GetAccessKeyLastUsed](#) IAM API リファレンスの
- [UpdateAccessKey](#) IAM API リファレンスの
- [DeleteAccessKey](#) IAM API リファレンスの

## IAM ユーザーを管理する

### ユーザーの作成

ユーザー名を含む [CreateUserRequest](#) オブジェクトを使用して、IamClientの createUserメソッドにユーザー名を指定IAMして、新しいユーザーを作成します。

### インポート

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;
```

## Code

```
public static String createIAMUser(IamClient iam, String username ) {

    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();

        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        CreateUserResponse response = iam.createUser(request);

        // Wait until the user is created
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user().userName();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

「」で[完全な例](#)を参照してください [GitHub](#)。

## ユーザーのリストを取得する

アカウントのIAMユーザーを一覧表示するには、新しい `IamClient` を作成し [ListUsersRequest](#)、それを `IamClient` の `listUsers` メソッドに渡します。返された [ListUsersResponse](#) オブジェクト `users` で呼び出すことで、ユーザーのリストを取得できます。

`listUsers` によって返されたユーザーのリストはページ分割されます。取得できる結果がさらにあることを確認するには、応答オブジェクトの `isTruncated` メソッドを呼び出します。true が返った場合は、応答オブジェクトの `marker()` メソッドを呼び出します。マーカー値を使用して、新しいリクエストオブジェクトを作成します。次に、`listUsers` メソッドを新しいリクエストで再度呼び出します。

### インポート

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.services.iam.model.User;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

### Code

```
public static void listAllUsers(IamClient iam ) {

    try {

        boolean done = false;
        String newMarker = null;

        while(!done) {
            ListUsersResponse response;

            if (newMarker == null) {
                ListUsersRequest request = ListUsersRequest.builder().build();
                response = iam.listUsers(request);
            } else {
                ListUsersRequest request = ListUsersRequest.builder()
                    .marker(newMarker).build();
                response = iam.listUsers(request);
            }
        }
    }
}
```

```
        for(User user : response.users()) {
            System.out.format("\n Retrieved user %s", user.userName());
        }

        if(!response.isTruncated()) {
            done = true;
        } else {
            newMarker = response.marker();
        }
    }
} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

「」で[完全な例](#)を参照してください GitHub。

## ユーザーの更新

ユーザーを更新するには、IamClient オブジェクトの updateUser メソッドを呼び出します。このメソッドは、ユーザー名またはパスを変更するために使用できる [UpdateUserRequest](#) オブジェクトを受け取ります。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;
```

### Code

```
public static void updateIAMUser(IamClient iam, String curName, String newName) {

    try {
        UpdateUserRequest request = UpdateUserRequest.builder()
            .userName(curName)
            .newUserName(newName)
            .build();

        iam.updateUser(request);
        System.out.printf("Successfully updated user to username %s",
```

```
        newName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

「」で[完全な例](#)を参照してください GitHub。

## ユーザーを削除する

ユーザーを削除するには、削除するユーザー名で設定された [UpdateUserRequest](#) オブジェクトを使用して iamClient の deleteUser リクエストを呼び出します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

### Code

```
public static void deleteIAMUser(IamClient iam, String userName) {

    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("Successfully deleted IAM user " + userName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

「」で[完全な例](#)を参照してください GitHub。

## 詳細情報

- 「IAM ユーザーガイド」の [IAM ユーザー](#)

- IAM ユーザーガイドの [IAM ユーザーの管理](#)
- [CreateUser](#) IAM API リファレンスの
- [ListUsers](#) IAM API リファレンスの
- [UpdateUser](#) IAM API リファレンスの
- [DeleteUser](#) IAM API リファレンスの

## を使用して IAM ポリシーを作成する AWS SDK for Java 2.x

[IAM Policy Builder API](#) は、Java で [IAM ポリシー](#) を構築し、AWS Identity and Access Management (IAM) にアップロードするために使用できるライブラリです。

この API では、JSON 文字列を手動で組み立てたりファイルを読み取ったりして IAM ポリシーを構築する代わりに、クライアント側のオブジェクト指向の方法で JSON 文字列を生成します。既存の IAM ポリシーを JSON 形式で読み取ると、API は処理のためにそれを [IamPolicy](#) インスタンスに変換します。

IAM Policy Builder API は SDK のバージョン 2.20.105 で利用できるようになったため、Maven ビルドファイルではそのバージョンまたはそれ以降のバージョンを使用してください。SDK の最新バージョン番号は [Maven central にリスト](#) があります。

次のスニペットは、Maven pom.xml ファイルの依存関係ブロック例を示しています。これにより、IAM Policy Builder API をプロジェクトで使用できるようになります。

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>iam-policy-builder</artifactId>
  <version>2.20.139</version>
</dependency>
```

### IamPolicy の作成

このセクションでは、IAM Policy Builder API を使用してポリシーを構築する方法の例をいくつか示します。

以下の各例では、[IamPolicy.Builder](#) から始めて、addStatement メソッドを使用して 1 つ以上のステートメントを追加します。このパターンに従って、[IamStatement.Builder](#) には、効果、アクション、リソース、および条件をステートメントに追加するメソッドがあります。

## 例: 時間ベースのポリシーを作成する

次の例では、2つの時点間の Amazon DynamoDB GetItem アクションを許可するアイデンティティベースのポリシーを作成します。

```
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 出力

前の例の最後のステートメントは、次の JSON 文字列を返します。

[この例](#)の詳細については、「AWS Identity and Access Management ユーザーガイド」を参照してください。

```
{
  "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"
  }
}
```

### 例: 複数の条件を指定する

次の例では、特定の DynamoDB 属性へのアクセスを許可する ID ベースポリシーを作成する方法を示します。ポリシーには 2 つの条件が含まれています。

```
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 出力

前の例の最後のステートメントは、次の JSON 文字列を返します。



[この例](#)の詳細については、「AWS Identity and Access Management ユーザーガイド」を参照してください。

```
{
  "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"
      }
    }
  }
}
```

例: プリンシパルを指定する

次の例は、条件で指定されたものを除くすべてのプリンシパルのバケットへのアクセスを拒否するリソーススペースのポリシーを作成する方法を示しています。

```
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 出力

前の例の最後のステートメントは、次の JSON 文字列を返します。

[この例](#)の詳細については、「AWS Identity and Access Management ユーザーガイド」を参照してください。

```
{
  "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"
      }
    }
  }
}
```

例: クロスアカウントアクセスを許可します。

次の例は、アップロードされたオブジェクトの完全な所有者制御を維持しながら AWS アカウント、別のバケットにオブジェクトをアップロードできるようにする方法を示しています。

```
public String allowCrossAccountAccessExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addPrincipal(IamPrincipalType.AWS, "111122223333")
            .addAction("s3:PutObject")
            .addResource("arn:aws:s3:::DOC-EXAMPLE-BUCKET/*")
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.STRING_EQUALS)
                .key("s3:x-amz-acl")
                .value("bucket-owner-full-control")))
        .build();
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}
```

## JSON 出力

前の例の最後のステートメントは、次の JSON 文字列を返します。

[この例](#)の詳細については、「Amazon Simple Storage Service ユーザーガイド」を参照ください。

```
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "111122223333"
    },
    "Action" : "s3:PutObject",
    "Resource" : "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
    "Condition" : {
      "StringEquals" : {
        "s3:x-amz-acl" : "bucket-owner-full-control"
      }
    }
  }
}
```

## IAM で **IamPolicy** を使用する

IamPolicy インスタンスを作成したら、[IamClient](#) を使用して IAM サービスを利用します。

次の例では、[IAM ID](#) が accountID パラメータで指定されたアカウントの DynamoDB テーブルに項目を書き込むことを許可するポリシーを構築します。次に、ポリシーが JSON 文字列として IAM にアップロードされます。

```
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());
}

```

次の例は、前述の例に基づいています。コードはポリシーをダウンロードし、ステートメントをコピーして変更することで新しいポリシーの基礎として使用します。その後、新しいポリシーがアップロードされます。

```

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

前の例では、次のスニペットで示される IamClient 引数を使用します。

```
IamClient iam = IamClient.builder().region(Region.AWS_GLOBAL).build();
```

## JSON のポリシー

例では次の JSON 文字列が返されます。

```
First example
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : "dynamodb:PutItem",
    "Resource" : "arn:aws:dynamodb:us-east-1:111122223333:table/exampleTableName"
  }
}

Second example
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : [ "dynamodb:PutItem", "dynamodb:GetItem" ],
    "Resource" : "arn:aws:dynamodb:us-east-1:111122223333:table/exampleTableName"
  }
}
```

## IAM ポリシーの使用

### ポリシーの作成

新しいポリシーを作成するには、の IamClientcreatePolicyメソッド [CreatePolicyRequest](#) にポリシーの名前と JSON 形式のポリシードキュメントを指定します。

## インポート

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
```

## Code

```
public static String createIAMPolicy(IamClient iam, String policyName ) {

    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();

        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument).build();

        CreatePolicyResponse response = iam.createPolicy(request);

        // Wait until the policy is created
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);
        waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}
```

「」で[完全な例](#)を参照してください GitHub。

## ポリシーの取得

既存のポリシーを取得するには、IamClientの getPolicyメソッドを呼び出し、[GetPolicyRequest](#) オブジェクト内でポリシーの ARN を指定します。

### インポート

```
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

### Code

```
public static void getIAMPolicy(IamClient iam, String policyArn) {

    try {
        GetPolicyRequest request = GetPolicyRequest.builder()
            .policyArn(policyArn).build();

        GetPolicyResponse response = iam.getPolicy(request);
        System.out.format("Successfully retrieved policy %s",
            response.policy().policyName());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

「」で[完全な例](#)を参照してください GitHub。

## ロールポリシーのアタッチ

の IamClientattachRolePolicyメソッドを呼び出し、のIAM[ロール](#)名とポリシー ARN を指定することで、ポリシーをロールにアタッチできます[AttachRolePolicyRequest](#)。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;
```

## Code

```
public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {

    try {

        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
        .roleName(roleName)
        .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        // Ensure that the policy is not attached to this role
        String polArn = "";
        for (AttachedPolicy policy: attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn)==0) {
                System.out.println(roleName +
                    " policy is already attached to this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
            AttachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policyArn)
                .build();

        iam.attachRolePolicy(attachRequest);
    }
}
```



```
        System.out.println("Successfully attached policy " + policyArn +
            " to role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("Done");
}
```

「」で[完全な例](#)を参照してください GitHub。

## アタッチ済みロールポリシーの一覧表示

IamClientの `listAttachedRolePolicies` メソッドを呼び出して、ロールにアタッチされたポリシーを一覧表示します。ポリシーを一覧表示するロール名を含む [ListAttachedRolePoliciesRequest](#) オブジェクトを受け取ります。

返された [ListAttachedRolePoliciesResponse](#) オブジェクト `getAttachedPolicies` を呼び出して、アタッチされたポリシーのリストを取得します。結果は切り捨てられる場合があります。 `ListAttachedRolePoliciesResponse` オブジェクトの `isTruncated` メソッドが `true` を返す場合は、 `ListAttachedRolePoliciesResponse` オブジェクトの `marker` メソッドを呼び出します。返るマーカーを使用して新しいリクエストを作成します。また、それを使用して `listAttachedRolePolicies` を再び呼び出し、結果の次のバッチを取得します。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;
```

## Code

```
public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {
```

```
try {

    ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
    .roleName(roleName)
    .build();

    ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
    List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

    // Ensure that the policy is not attached to this role
    String polArn = "";
    for (AttachedPolicy policy: attachedPolicies) {
        polArn = policy.policyArn();
        if (polArn.compareTo(policyArn)==0) {
            System.out.println(roleName +
                " policy is already attached to this role.");
            return;
        }
    }

    AttachRolePolicyRequest attachRequest =
        AttachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

    iam.attachRolePolicy(attachRequest);

    System.out.println("Successfully attached policy " + policyArn +
        " to role " + roleName);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Done");
}
```

「」で[完全な例](#)を参照してください GitHub。

## ロールポリシーのデタッチ

ロールからポリシーをデタッチするには、IamClientの detachRolePolicyメソッドを呼び出し、でロール名とポリシー ARN を指定します [DetachRolePolicyRequest](#)。

### インポート

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

### Code

```
public static void detachPolicy(IamClient iam, String roleName, String policyArn )
{
    try {
        DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.detachRolePolicy(request);
        System.out.println("Successfully detached policy " + policyArn +
            " from role " + roleName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

「」で [完全な例](#) を参照してください GitHub。

## 詳細情報

- IAM ユーザーガイドの [IAM ポリシーの概要](#)。
- IAM ユーザーガイドの [AWS IAM ポリシーのリファレンス](#)。
- [CreatePolicy](#) IAM API リファレンスの
- [GetPolicy](#) IAM API リファレンスの

- [AttachRolePolicy](#) IAM API リファレンスの
- [ListAttachedRolePolicies](#) IAM API リファレンスの
- [DetachRolePolicy](#) IAM API リファレンスの

## IAM サーバー証明書の使用

でウェブサイトまたはアプリケーションへの HTTPS 接続を有効にするには AWS、SSL/TLS サーバー証明書が必要です。が提供するサーバー証明書、AWS Certificate Manager または外部プロバイダーから取得したサーバー証明書を使用できます。

ACM を使用してサーバー証明書のプロビジョニング、管理、デプロイを行うことをお勧めします。ACM を使用すると、証明書をリクエストし、AWS リソースにデプロイして、が証明書の更新 ACM を処理するようにできます。が提供する証明書 ACM は無料です。の詳細については ACM、「[AWS Certificate Manager ユーザーガイド](#)」を参照してください。

### サーバー証明書の取得

IamClient の `getServerCertificate` メソッドを呼び出し、証明書の名前 [GetServerCertificateRequest](#) で を渡すことで、サーバー証明書を取得できます。

### インポート

```
import software.amazon.awssdk.services.iam.model.GetServerCertificateRequest;
import software.amazon.awssdk.services.iam.model.GetServerCertificateResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

### Code

```
public static void getCertificate(IamClient iam, String certName ) {

    try {
        GetServerCertificateRequest request = GetServerCertificateRequest.builder()
            .serverCertificateName(certName)
            .build();

        GetServerCertificateResponse response = iam.getServerCertificate(request);
        System.out.format("Successfully retrieved certificate with body %s",
            response.serverCertificate().certificateBody());
    }
}
```

```
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

「」で[完全な例](#)を参照してください GitHub。

## サーバー証明書の一覧表示

サーバー証明書を一覧表示するには、IamClientの listServerCertificates メソッドを で呼び出します [ListServerCertificatesRequest](#)。このメソッドからは、[ListServerCertificatesResponse](#) が返されます。

返された ListServerCertificateResponse オブジェクトの serverCertificateMetadataList メソッドを呼び出して、各証明書に関する情報を取得するために使用できる [ServerCertificateMetadata](#) オブジェクトのリストを取得します。

結果は切り捨てられる場合があります。ListServerCertificateResponse オブジェクトの isTruncated メソッドが true を返す場合は、ListServerCertificatesResponse オブジェクトの marker メソッドを呼び出します。その後、マーカーを使用して新しいリクエストを作成します。この新しいリクエストを使用して、再度 listServerCertificates を呼び出し、結果の次のバッチを取得します。

## インポート

```
import software.amazon.awssdk.services.iam.model.IamException;  
import software.amazon.awssdk.services.iam.model.ListServerCertificatesRequest;  
import software.amazon.awssdk.services.iam.model.ListServerCertificatesResponse;  
import software.amazon.awssdk.services.iam.model.ServerCertificateMetadata;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.iam.IamClient;
```

## Code

```
public static void listCertificates(IamClient iam) {  
  
    try {  
        boolean done = false;  
        String newMarker = null;
```

```
while(!done) {
    ListServerCertificatesResponse response;

    if (newMarker == null) {
        ListServerCertificatesRequest request =
            ListServerCertificatesRequest.builder().build();
        response = iam.listServerCertificates(request);
    } else {
        ListServerCertificatesRequest request =
            ListServerCertificatesRequest.builder()
                .marker(newMarker).build();
        response = iam.listServerCertificates(request);
    }

    for(ServerCertificateMetadata metadata :
        response.serverCertificateMetadataList()) {
        System.out.printf("Retrieved server certificate %s",
            metadata.serverCertificateName());
    }

    if(!response.isTruncated()) {
        done = true;
    } else {
        newMarker = response.marker();
    }
}

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

「」で[完全な例](#)を参照してください GitHub。

## サーバー証明書の更新

の `IamClient.updateServerCertificate` メソッドを呼び出すことで、サーバー証明書の名前またはパスを更新できます。サーバー証明書の現在の名前と、使用する新しい名前または新しいパスを持つ [UpdateServerCertificateRequest](#) オブジェクトセットを受け取ります。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateServerCertificateRequest;
import software.amazon.awssdk.services.iam.model.UpdateServerCertificateResponse;
```

## Code

```
public static void updateCertificate(IamClient iam, String curName, String newName)
{
    try {
        UpdateServerCertificateRequest request =
            UpdateServerCertificateRequest.builder()
                .serverCertificateName(curName)
                .newServerCertificateName(newName)
                .build();

        UpdateServerCertificateResponse response =
            iam.updateServerCertificate(request);

        System.out.printf("Successfully updated server certificate to name %s",
            newName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

「」で[完全な例](#)を参照してください [GitHub](#)。

## サーバー証明書の削除

サーバー証明書を削除するには、証明書の名前 [DeleteServerCertificateRequest](#) を含む `iamClient.deleteServerCertificate` メソッドを呼び出します。

## インポート

```
import software.amazon.awssdk.services.iam.model.DeleteServerCertificateRequest;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

## Code

```
public static void deleteCert(IamClient iam,String certName ) {

    try {
        DeleteServerCertificateRequest request =
            DeleteServerCertificateRequest.builder()
                .serverCertificateName(certName)
                .build();

        iam.deleteServerCertificate(request);
        System.out.println("Successfully deleted server certificate " +
            certName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

「」で[完全な例](#)を参照してください GitHub。

## 詳細情報

- IAM ユーザーガイドの[サーバー証明書の使用](#)
- [GetServerCertificate](#) IAM API リファレンスの
- [ListServerCertificates](#) IAM API リファレンスの
- [UpdateServerCertificate](#) IAM API リファレンスの
- [DeleteServerCertificate](#) IAM API リファレンスの
- [AWS Certificate Manager ユーザーガイド](#)

## の操作 Kinesis

このセクションでは、AWS SDK for Java 2.x [Amazon Kinesis](#)を使用したプログラミングの例を示します。



の詳細については Kinesis、[「Amazon Kinesis デベロッパーガイド」](#)を参照してください。

次の例には各手法を示すのに必要なコードのみが含まれます。[完全なサンプルコードは、「」で入手できます GitHub](#)。そこから、単一のソースファイルをダウンロードするカリポジトリをローカルにクローン作成して、ビルドし実行するためのすべての例を取得できます。

## トピック

- [Amazon Kinesis Data Streams へのサブスクライブ](#)

## Amazon Kinesis Data Streams へのサブスクライブ

以下の例では、Amazon Kinesis メソッドを使用し、subscribeToShard Data Streams からデータを取得して処理する方法について説明します。Kinesis Data Streams では、強化されたファンアウト機能と低レイテンシーの HTTP/2 データ取得 API が導入されました。これにより、デベロッパーは複数の低レイテンシーで高パフォーマンスのアプリケーションを、同じ Kinesis Data Streams でより簡単に実行できます。

## セットアップする

まず、非同期 Kinesisクライアントと [SubscribeToShardRequest](#) オブジェクトを作成します。これらのオブジェクトは、Kinesis イベントにサブスクライブする次のそれぞれの例で使用されます。

## インポート

```
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.atomic.AtomicInteger;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;
import software.amazon.awssdk.services.kinesis.model.ShardIteratorType;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEvent;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEventStream;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardRequest;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponse;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponseHandler;
```

## Code

```
Region region = Region.US_EAST_1;
```

```
KinesisAsyncClient client = KinesisAsyncClient.builder()
    .region(region)
    .build();

SubscribeToShardRequest request = SubscribeToShardRequest.builder()
    .consumerARN(CONSUMER_ARN)
    .shardId("arn:aws:kinesis:us-east-1:111122223333:stream/
StockTradeStream")
    .startingPosition(s -> s.type(ShardIteratorType.LATEST)).build();
```

## ビルダーインターフェイスを使用する

builder メソッドを使用して、[SubscribeToShardResponseHandler](#) の作成をシンプルにできます。

ビルダーを使用して、完全なインターフェイスを実装する代わりにメソッド呼び出しで各ライフサイクルのコールバックを設定できます。

### Code

```
private static CompletableFuture<Void> responseHandlerBuilder(KinesisAsyncClient
client, SubscribeToShardRequest request) {
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .onComplete(() -> System.out.println("All records stream
successfully"))
    // Must supply some type of subscriber
    .subscriber(e -> System.out.println("Received event - " + e))
    .build();
    return client.subscribeToShard(request, responseHandler);
}
```

公開者をさらに制御するには、publisherTransformer メソッドを使用して公開者をカスタマイズできます。

### Code

```
private static CompletableFuture<Void>
responseHandlerBuilderPublisherTransformer(KinesisAsyncClient client,
SubscribeToShardRequest request) {
```

```
SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .publisherTransformer(p -> p.filter(e -> e instanceof
SubscribeToShardEvent).limit(100))
    .subscriber(e -> System.out.println("Received event - " + e))
    .build();
return client.subscribeToShard(request, responseHandler);
}
```

[GitHub](#) で完全な例をご覧ください。

## カスタムレスポンスハンドラを使用する

サブスクライバーとパブリッシャーの完全なコントロールのために、SubscribeToShardResponseHandler インターフェイスを実装します。

この例では、onEventStream メソッドを実装します。これにより、公開者へのフルアクセスが許可されます。このデモでは、受信者による表示のために公開者をイベントレコードに変換する方法を示します。

### Code

```
private static CompletableFuture<Void>
responseHandlerBuilderClassic(KinesisAsyncClient client, SubscribeToShardRequest
request) {
    SubscribeToShardResponseHandler responseHandler = new
SubscribeToShardResponseHandler() {

        @Override
        public void responseReceived(SubscribeToShardResponse response) {
            System.out.println("Receieved initial response");
        }

        @Override
        public void onEventStream(SdkPublisher<SubscribeToShardEventStream>
publisher) {
            publisher
                // Filter to only SubscribeToShardEvents
                .filter(SubscribeToShardEvent.class)
                // Flat map into a publisher of just records
            ;
        }
    };
}
```

```

        .flatMapIterable(SubscribeToShardEvent::records)
        // Limit to 1000 total records
        .limit(1000)
        // Batch records into lists of 25
        .buffer(25)
        // Print out each record batch
        .subscribe(batch -> System.out.println("Record Batch - " +
batch));
    }

    @Override
    public void complete() {
        System.out.println("All records stream successfully");
    }

    @Override
    public void exceptionOccurred(Throwable throwable) {
        System.err.println("Error during stream - " + throwable.getMessage());
    }
};
return client.subscribeToShard(request, responseHandler);
}

```

[GitHub](#) で完全な例をご覧ください。

## 訪問者インターフェイスを使用する

[Visitor](#) オブジェクトを使用して、監視する特定のイベントにサブスクライブできます。

### Code

```

private static CompletableFuture<Void>
responseHandlerBuilderVisitorBuilder(KinesisAsyncClient client,
SubscribeToShardRequest request) {
    SubscribeToShardResponseHandler.Visitor visitor =
SubscribeToShardResponseHandler.Visitor
        .builder()
        .onSubscribeToShardEvent(e -> System.out.println("Received subscribe to
shard event " + e))
        .build();
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
        .builder()

```

```
        .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
        .subscriber(visitor)
        .build();
return client.subscribeToShard(request, responseHandler);
}
```

[GitHub](#) で完全な例をご覧ください。

## カスタム受信者を使用する

独自のカスタム受信者を実装して、ストリームにサブスクライブすることもできます。

このコードスニペットでは、サブスクライバーの例を示しています。

### Code

```
private static class MySubscriber implements
Subscriber<SubscribeToShardEventStream> {

    private Subscription subscription;
    private AtomicInteger eventCount = new AtomicInteger(0);

    @Override
    public void onSubscribe(Subscription subscription) {
        this.subscription = subscription;
        this.subscription.request(1);
    }

    @Override
    public void onNext(SubscribeToShardEventStream shardSubscriptionEventStream) {
        System.out.println("Received event " + shardSubscriptionEventStream);
        if (eventCount.incrementAndGet() >= 100) {
            // You can cancel the subscription at any time if you wish to stop
receiving events.
            subscription.cancel();
        }
        subscription.request(1);
    }

    @Override
    public void onError(Throwable throwable) {
        System.err.println("Error occurred while stream - " +
throwable.getMessage());
    }
}
```

```
    }

    @Override
    public void onComplete() {
        System.out.println("Finished streaming all events");
    }
}
```

次のコードスニペットに示すように、カスタムサブスクライバーを `subscribe` メソッドに渡すことができます。

#### Code

```
private static CompletableFuture<Void>
responseHandlerBuilderSubscriber(KinesisAsyncClient client, SubscribeToShardRequest
request) {
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .subscriber(MySubscriber::new)
    .build();
    return client.subscribeToShard(request, responseHandler);
}
```

[GitHub](#) で完全な例をご覧ください。

## Kinesis データストリームにデータレコードを書き込む

[KinesisClient](#) オブジェクトで `putRecords` メソッドを使用して、データレコードを Kinesis データストリームに書き込むことができます。このメソッドを正常に呼び出すには、[PutRecordsRequest](#) オブジェクトを作成します。データストリームの名前を `streamName` メソッドに渡します。また、`putRecords` メソッドを使用してデータを渡す必要があります (次のコード例に示します)。

### インポート

```
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;
```

次の Java コード例では、StockTrade オブジェクトが Kinesis データストリームに書き込むデータとして使用されています。この例を実行する前に、データストリームを作成したことを確認します。

## Code

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StockTradesWriter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <streamName>

            Where:
                streamName - The Amazon Kinesis data stream to which records are
written (for example, StockTradeStream)
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
```

```
        .region(region)
        .build();

// Ensure that the Kinesis Stream is valid.
validateStream(kinesisClient, streamName);
setStockData(kinesisClient, streamName);
kinesisClient.close();
}

public static void setStockData(KinesisClient kinesisClient, String streamName) {
    try {
        // Repeatedly send stock trades with a 100 milliseconds wait in between.
        StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

        // Put in 50 Records for this example.
        int index = 50;
        for (int x = 0; x < index; x++) {
            StockTrade trade = stockTradeGenerator.getRandomTrade();
            sendStockTrade(trade, kinesisClient, streamName);
            Thread.sleep(100);
        }

    } catch (KinesisException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done");
}

private static void sendStockTrade(StockTrade trade, KinesisClient kinesisClient,
    String streamName) {
    byte[] bytes = trade.toJsonAsBytes();

    // The bytes could be null if there is an issue with the JSON serialization by
    // the Jackson JSON library.
    if (bytes == null) {
        System.out.println("Could not get JSON bytes for stock trade");
        return;
    }

    System.out.println("Putting trade: " + trade);
    PutRecordRequest request = PutRecordRequest.builder()
        .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol as
the partition key, explained in
```



```

                                                                    // the Supplemental Information
section below.
        .streamName(streamName)
        .data(SdkBytes.fromByteArray(bytes))
        .build();

    try {
        kinesisClient.putRecord(request);
    } catch (KinesisException e) {
        System.err.println(e.getMessage());
    }
}

private static void validateStream(KinesisClient kinesisClient, String streamName)
{
    try {
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
            .streamName(streamName)
            .build();

        DescribeStreamResponse describeStreamResponse =
kinesisClient.describeStream(describeStreamRequest);

        if (!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
        {
            System.err.println("Stream " + streamName + " is not active. Please
wait a few moments and try again.");
            System.exit(1);
        }

    } catch (KinesisException e) {
        System.err.println("Error found while describing the stream " +
streamName);
        System.err.println(e);
        System.exit(1);
    }
}
}
```

[GitHub](#) で完全な例をご覧ください。

## サードパーティーライブラリを使用する

カスタムの受信者を実装せずに、その他のサードパーティーのライブラリを使用することができます。この例では、RxJava の実装を例に挙げていますが、リアクティブなストリームのインターフェイスを実装するライブラリを使用することもできます。上記ライブラリの詳細については、「[Github の RxJava wiki ページ](#)」を参照してください。

このライブラリを使用するには、依存関係として追加します。使用する POM スニペットの例を示します (Maven を使用している場合)。

### POM エントリ

```
<dependency>
  <groupId>io.reactivex.rxjava2</groupId>
  <artifactId>rxjava</artifactId>
  <version>2.1.14</version>
</dependency>
```

### インポート

```
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;
```

この例では、onEventStream ライフサイクルメソッドの RxJava を使用します。これにより、発行者へのフルアクセスが付与され、これを使用して Rx Flowable を作成できます。

### Code

```
SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .onEventStream(p -> Flowable.fromPublisher(p)
        .ofType(SubscribeToShardEvent.class)

.flatMapIterable(SubscribeToShardEvent::records)
        .limit(1000)
        .buffer(25)
        .subscribe(e -> System.out.println("Record
batch = " + e)))
    .build();
```

`publisherTransformer` 発行者を指定して、`Flowable` メソッドを使用することもできます。次の例に示すように、`Flowable` 発行者を `SdkPublisher` に適用する必要があります。

## Code

```
SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .publisherTransformer(p ->
SdkPublisher.adapt(Flowable.fromPublisher(p).limit(100)))
    .build();
```

[GitHub](#) で完全な例をご覧ください。

## 詳細情報

- Amazon Kinesis API リファレンスの [SubscribeToShardEvent](#)
- Amazon Kinesis API リファレンスの [SubscribeToShard](#)

## AWS Lambda 関数の呼び出し、一覧表示、および削除

このセクションでは、AWS SDK for Java 2.x を使用して Lambda サービスクライアントでプログラミングする例を示します。

## トピック

- [Lambda 関数を呼び出します。](#)
- [Lambda 関数を一覧表示する](#)
- [Lambda 関数を削除する](#)

## Lambda 関数を呼び出します。

オブジェクトを作成し、その `invoke` メソッドを呼び [LambdaClient](#) 出すことで、Lambda 関数を呼び出すことができます。 [InvokeRequest](#) オブジェクトを作成して、関数名や関数に渡すペイロードなどの追加情報を指定します Lambda。関数名は `arn:aws:lambda:us-east-1:123456789012:function:HelloFunction` と表示されます。AWS Management Console で関数を確認することで、値を取得できます。

ペイロードデータを関数に渡すには、情報を含む [SdkBytes](#) オブジェクトを作成します。たとえば、次のコード例では、Lambda 関数に渡される JSON データに注意してください。

### インポート

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.InvokeRequest;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lambda.model.InvokeResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;
```

### Code

次のコード例は、Lambda 関数を呼び出す方法を示しています。

```
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);
}
}
```

「」で[完全な例](#)を参照してください GitHub。

## Lambda 関数を一覧表示する

[Lambda Client](#) オブジェクトを構築し、その `listFunctions` メソッドを呼び出します。このメソッドは [ListFunctionsResponse](#) オブジェクトを返します。このオブジェクトの `functions` メソッドを呼び出して、[FunctionConfiguration](#) オブジェクトのリストを返すことができます。リストを反復処理して、関数に関する情報を取得できます。たとえば、次の Java コード例は、各関数名を取得する方法を示しています。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.ListFunctionsResponse;
import software.amazon.awssdk.services.lambda.model.FunctionConfiguration;
import java.util.List;
```

### Code

次の Java コード例は、関数名のリストを取得する方法を示しています。

```
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);
    }
}
```

「」で[完全な例](#)を参照してください GitHub。

## Lambda 関数を削除する

[LambdaClient](#) オブジェクトを構築し、その `deleteFunction` メソッドを呼び出します。 [DeleteFunctionRequest](#) オブジェクトを作成し、`deleteFunction` メソッドに渡します。このオブジェクトには、削除する関数の名前などの情報が含まれています。関数名は `arn:aws:lambda:us-east-1:123456789012:function:HelloFunction` と表示されます。AWS Management Console で関数を確認することで、値を取得できます。

### インポート

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.DeleteFunctionRequest;
import software.amazon.awssdk.services.lambda.model.LambdaException;
```

### Code

次の Java コードは、Lambda 関数を削除する方法を示しています。

```
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);
    }
}
```

「」で[完全な例](#)を参照してください GitHub。

## Amazon S3 で使用する

このセクションでは、AWS SDK for Java 2.xを使用して [Amazon Simple Storage Service \(S3\)](#) でプログラムする例を示します。

次の例には各手法を示すのに必要なコードのみが含まれます。[完全なサンプルコードは、「」で入手できます GitHub](#)。そこから、単一のソースファイルをダウンロードするカリポジトリをローカルにクローン作成して、ビルドし実行するためのすべての例を取得できます。

### Note

バージョン 2.18.x 以降では、はエンドポイントオーバーライドを含めるときに仮想ホスト形式のアドレス指定 AWS SDK for Java 2.x を使用します。これは、バケット名が有効な DNS ラベルである限り適用されます。

クライアントビルダーの `true` で [forcePathStyle](#) メソッドを呼び出して、クライアントにバケットにパス形式のアドレス指定を使用するように強制します。

以下の例は、エンドポイントオーバーライドを設定し、パス形式のアドレス指定を使用するサービスクライアントを示しています。

```
S3Client client = S3Client.builder()
    .region(Region.US_WEST_2)
    .endpointOverride(URI.create("https://s3.us-
west-2.amazonaws.com"))
    .forcePathStyle(true)
    .build();
```

## アクセスポイントまたはマルチリージョンアクセスポイントを使用する

[Amazon S3 Access Points](#) または [マルチリージョンアクセスポイント](#) を設定したら、`putObject` や `getObject` などのオブジェクトメソッドを呼び出して、バケット名の代わりにアクセスポイント ID を指定できます。

たとえば、アクセスポイント ARN 識別子が `arn:aws:s3:us-west-2:123456789012:accesspoint/test` の場合、次のスニペットを使用して `putObject` メソッドを呼び出すことができます。

```
Path path = Paths.get(URI.create("file:///temp/file.txt"));

s3Client.putObject(builder -> builder
    .key("myKey")
    .bucket("arn:aws:s3:us-west-2:123456789012:accesspoint/test")
    , path);
```

ARN 文字列の代わりに、アクセスポイントの [バケットスタイルのエイリアス](#) を `bucket` パラメータに使用することもできます。

マルチリージョンアクセスポイントを使用するには、`bucket` パラメータを次の形式のマルチリージョンアクセスポイント ARN に置き換えます。

```
arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias
```

SDK for Java を使用してマルチリージョンアクセスポイントと連携するには、次の Maven 依存関係を追加します。Maven central で [最新バージョン](#) を検索します。

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>auth-crt</artifactId>
  <version>VERSION</version>
</dependency>
```

## トピック

- [Amazon S3 バケットの作成、一覧表示および削除](#)
- [Amazon S3 オブジェクトの操作](#)
- [Amazon S3 の署名付き URL を操作する](#)
- [Amazon S3 のクロスリージョンアクセス](#)
- [による Amazon S3 チェックサム](#)
- [パフォーマンスの高い S3 クライアントを使用する: AWS CRT ベースの S3 クライアント](#)
- [Amazon S3 Transfer Manager を使用してファイルとディレクトリを転送する](#)



## Amazon S3 バケットの作成、一覧表示および削除

Amazon S3 の各オブジェクト (ファイル) がバケット内に存在している必要があります。バケットは、オブジェクトの集合 (コンテナ) を表します。各バケットには、一意のキー (名前) が必要です。バケットおよびその設定についての詳細は、Amazon Simple Storage Service ユーザーガイドの [Amazon S3 バケットの使用](#) を参照してください。

### Note

ベストプラクティス

[バケットで AbortIncompleteMultipartUpload](#) Amazon S3 ライフサイクルルールを有効にすることをお勧めします。

このルールは、開始後、指定された日数内に完了しないマルチパートアップロードを中止するよう Amazon S3 に指示できます。設定した時間制限を超えると、Amazon S3 はアップロードを中止して、不完全なアップロードデータを削除します。

詳細については、Amazon Simple Storage Service ユーザーガイドの [バージョンニングを使用したバケットのライフサイクル設定](#) を参照してください。

### Note

これらのコードスニペットは、ユーザーが基本的な内容を理解しており、かつ、[the section called “SDK のシングルサインオンアクセスのセットアップ”](#) の情報を使用してデフォルトの AWS 認証情報を設定していることを前提としています。

## バケットを作成する

バケット名を含む [CreateBucketRequest](#) を作成します。S3Client の createBucket メソッドに渡します。バケットの一覧表示または削除などのオペレーションを行うには、後の例に示すように、S3Client を使用します。

### インポート

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

## Code

最初に、S3Client を作成します。

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

バケットの作成リクエストを作成します。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3BucketOps {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
```

```
        .region(region)
        .build();

String bucket = "bucket" + System.currentTimeMillis();
System.out.println(bucket);
createBucket(s3, bucket);
performOperations(s3, bucket);
}

// Create a bucket by using a S3Waiter object
public static void createBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

[GitHub](#) で完全な例をご覧ください。

## バケットの一覧表示

[ListBucketsRequest](#) を作成します。S3Client の listBuckets メソッドを使用して、バケットのリストを取得します。リクエストが成功すると、[ListBucketsResponse](#) が返されます。この応答オブジェクトを使用して、バケットのリストを取得します。

## インポート

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

## Code

最初に、S3Client を作成します。

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

バケットの一覧表示リクエストを作成します。

```
// List buckets
ListBucketsRequest listBucketsRequest = ListBucketsRequest.builder().build();
ListBucketsResponse listBucketsResponse = s3.listBuckets(listBucketsRequest);
listBucketsResponse.buckets().stream().forEach(x ->
System.out.println(x.name()));
```

[GitHub](#) で完全な例をご覧ください。

## バケットを削除する

Amazon S3 バケットを削除する前に、バケットが空であることを必ず確認してください。空でない場合は、エラーが返ります。[バージョンングされたバケット](#)がある場合は、このバケット内にあるバージョンングされたオブジェクトも削除する必要があります。

### トピック

- [バケット内のオブジェクトを削除する](#)
- [空のバケットを削除する](#)

## バケット内のオブジェクトを削除する

[ListObjectsV2Request](#) を構築後、S3Client の `listObjects` メソッドを使用して、バケット内にあるオブジェクトのリストを取得します。次に、オブジェクトごとに `deleteObject` メソッドを使用して削除します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
```

### Code

最初に、S3Client を作成します。

```
ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .credentialsProvider(credentialsProvider)
    .build();
```

バケット内のすべてのオブジェクトを削除します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class S3BucketDeletion {
    public static void main(String[] args) throws Exception {
        final String usage = ""

            Usage:
                <bucket>

            Where:
                bucket - The bucket to delete (for example, bucket1).\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteObjectsInBucket(s3, bucket);
        s3.close();
    }

    public static void deleteObjectsInBucket(S3Client s3, String bucket) {
        try {
            // To delete a bucket, all the objects in the bucket must be deleted first.
            ListObjectsV2Request listObjectsV2Request = ListObjectsV2Request.builder()
                .bucket(bucket)
                .build();
            ListObjectsV2Response listObjectsV2Response;

            do {
                listObjectsV2Response = s3.listObjectsV2(listObjectsV2Request);
                for (S3Object s3Object : listObjectsV2Response.contents()) {
                    DeleteObjectRequest request = DeleteObjectRequest.builder()
                        .bucket(bucket)

```

```
                .key(s3object.key())
                .build();
            s3.deleteObject(request);
        }
    } while (listObjectsV2Response.isTruncated());
    DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder().bucket(bucket).build();
    s3.deleteBucket(deleteBucketRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

[GitHub](#) で完全な例をご覧ください。

## 空のバケットを削除する

バケット名を指定して [DeleteBucketRequest](#) を構築し、S3Client の deleteBucket メソッドに渡します。

## インポート

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

## Code

最初に、S3Client を作成します。

```
Region region = Region.US_EAST_1;
```

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

バケットを削除します。

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();

s3.deleteBucket(deleteBucketRequest);
s3.close();
```

[GitHub](#) で完全な例をご覧ください。

## Amazon S3 オブジェクトの操作

Amazon S3 オブジェクトは、ファイルまたはデータの集合を表します。オブジェクトはすべて、[バケット](#)に格納する必要があります。

### Note

ベストプラクティス

[バケット](#)で `AbortIncompleteMultipartUpload` Amazon S3 ライフサイクルルールを有効にすることをお勧めします。

このルールは、開始後、指定された日数内に完了しないマルチパートアップロードを中止するよう Amazon S3 に指示できます。設定した時間制限を超えると、Amazon S3 はアップロードを中止して、不完全なアップロードデータを削除します。

詳細については、Amazon Simple Storage Service ユーザーガイドの[バージョンニングを使用したバケットのライフサイクル設定](#)を参照してください。

### Note

これらのコードスニペットは、ユーザーが基本的な内容を理解しており、かつ、[the section called “SDK のシングルサインオンアクセスのセットアップ”](#) の情報を使用してデフォルトの AWS 認証情報を設定していることを前提としています。



## トピック

- [オブジェクトのアップロード](#)
- [複数のパートに分けてオブジェクトをアップロードする](#)
- [オブジェクトの削除](#)
- [オブジェクトのリスト化](#)
- [その他の例](#)

## オブジェクトのアップロード

バケット名とキー名を含む [PutObjectRequest](#) を作成します。次に、このオブジェクトのコンテンツを含む [RequestBody](#) と、PutObjectRequest オブジェクトを指定して、S3Client の putObject メソッドを使用します。バケットが存在している必要があり、存在しない場合はエラーが返ります。

### インポート

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

```
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

## Code

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

createBucket(s3, bucketName, region);

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

s3.putObject(objectRequest,
    RequestBody.fromByteBuffer(getRandomByteBuffer(10_000)));
```

[GitHub](#) で完全な例をご覧ください。

## 複数のパートに分けてオブジェクトをアップロードする

S3Client の createMultipartUpload メソッドを使用してアップロード ID を取得する次に、uploadPart メソッドを使用して、各パートをアップロードします。最後に、S3Client の completeMultipartUpload メソッドを使用して、アップロードされたすべてのパーツをマージし、アップロード操作を終了するよう Amazon S3 に指示します。

## インポート

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
```

```
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

## Code

```
        // First create a multipart upload and get the upload id
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = response.uploadId();
        System.out.println(uploadId);

        // Upload all the different parts of the object
        UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .partNumber(1).build();

        String etag1 = s3
            .uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
            .eTag();
```

```
CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
    .uploadId(uploadId)
    .partNumber(2).build();

String etag2 = s3
    .uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB)))
    .eTag();

CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

// Finally call completeMultipartUpload operation to tell S3 to merge
all
// uploaded
// parts and finish the multipart operation.
CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
    .parts(part1, part2)
    .build();

CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .multipartUpload(completedMultipartUpload)
    .build();

s3.completeMultipartUpload(completeMultipartUploadRequest);
```

[GitHub](#) で完全な例をご覧ください。

## オブジェクトの削除

バケット名とキー名を含む [DeleteObjectRequest](#) を作成します。S3Client の deleteObject メソッドを使用して、削除するバケットの名前とオブジェクトを渡します。指定されたバケットとオブジェクトキーが存在している必要があり、存在しない場合はエラーが返ります。

## インポート

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

## Code

```
DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
    .bucket(bucketName)
    .key(key)
    .build();

s3.deleteObject(deleteObjectRequest);
```

[GitHub](#) で完全な例をご覧ください。

## オブジェクトのコピー

[CopyObjectRequest](#) を作成し、オブジェクトがコピーされるバケット名、URL エンコードされた文字列値 (`URLEncoder.encode` メソッドを参照)、およびオブジェクトのキー名を指定しま

す。S3Client の copyObject メソッドを使用して、[CopyObjectRequest](#) オブジェクトを渡します。指定されたバケットとオブジェクトキーが存在している必要があり、存在しない場合はエラーが返ります。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

## Code

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CopyObject {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <objectKey> <fromBucket> <toBucket>

            Where:
                objectKey - The name of the object (for example, book.pdf).
                fromBucket - The S3 bucket name that contains the object (for
                example, bucket1).
                toBucket - The S3 bucket to copy the object to (for example,
                bucket2).

            """;
```

```
    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String objectKey = args[0];
    String fromBucket = args[1];
    String toBucket = args[2];
    System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    copyBucketObject(s3, fromBucket, objectKey, toBucket);
    s3.close();
}

public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .sourceBucket(fromBucket)
        .sourceKey(objectKey)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        return copyRes.copyObjectResult().toString();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

[GitHub](#) で完全な例をご覧ください。

## オブジェクトのリスト化

[ListObjectsRequest](#) を作成し、バケット名を指定します。次に、S3Client の `listObjects` メソッドを呼び出して、`ListObjectsRequest` オブジェクトを渡します。このメソッドは、バケット内のすべてのオブジェクトを含む [ListObjectsResponse](#) を返します。このオブジェクトの `contents` メソッドを呼び出して、オブジェクトのリストを取得できます。次のコード例に示すように、このリストを反復処理してオブジェクトを表示できます。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;
```

### Code

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListObjects {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
```



```
        <bucketName>\s

        Where:
            bucketName - The Amazon S3 bucket from which objects are read.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    listBucketObjects(s3, bucketName);
    s3.close();
}

public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.print("\n The name of the key is " + myValue.key());
            System.out.print("\n The object is " + calKb(myValue.size()) + " KBs");
            System.out.print("\n The owner is " + myValue.owner());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// convert bytes to kbs.
private static long calKb(Long val) {
```

```
        return val / 1024;
    }
}
```

[GitHub](#) で完全な例をご覧ください。

## その他の例

このガイドの「[コード例](#)」セクションには、[オブジェクトのダウンロード](#)方法など、Amazon S3 オブジェクトの操作例が他にも記載されています。

## Amazon S3 の署名付き URL を操作する

署名付き URL を使用すると、ユーザーに AWS 認証情報やアクセス許可がなくても、プライベート S3 オブジェクトに一時的にアクセスできます。

たとえば、Alice に S3 オブジェクトへのアクセス権限があり、そのオブジェクトへのアクセス権限を一時的に Bob と共有するとします。Alice は、署名付き Get リクエストを生成して Bob との共有し、Bob が Alice の認証情報にアクセスすることなくそのオブジェクトをダウンロードできるようにすることができます。HTTP GET リクエストと HTTP PUT リクエスト用の署名付き URL を生成できます。

オブジェクトの署名付き URL を生成して、オブジェクトをダウンロードします (GET リクエスト)。

以下の例は次の 2 つの部分で構成されます。

- パート 1: Alice はオブジェクトの署名付き URL を生成します。
- パート 2: Bob は署名済み URL を使用してオブジェクトをダウンロードします。

### パート 1: URL を生成する

Alice はすでに S3 バケットにオブジェクトを持っています。彼女は次のコードを使用して、Bob が後続の GET リクエストで使用できる URL 文字列を生成します。

### インポート

```
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();
    }
}
```

## パート 2: オブジェクトをダウンロードする

ボブは次の 3 つのコードオプションのいずれかを使用してオブジェクトをダウンロードします。または、ブラウザを使用して GET リクエストを実行することもできます。

### JDK `HttpURLConnection` を使用する (v1.1 以降)

```
/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
    Capture the response body to a byte array.

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setRequestMethod("GET");
        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}
```

### JDK `HttpClient` を使用する (v11 以降)

```
/* Use the JDK HttpClient (since v11) class to do the download. */
public byte[] useHttpClientToGet(String presignedUrlString) {
```

```
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
    Capture the response body to a byte array.

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpResponse<InputStream> response = httpClient.send(requestBuilder
            .uri(presignedUrl.toURI())
            .GET()
            .build(),
            HttpResponse.BodyHandlers.ofInputStream());

        IoUtils.copy(response.body(), byteArrayOutputStream);

        logger.info("HTTP response code is " + response.statusCode());
    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}
```

## SDK for Java から **SdkHttpClient** を使用する

```
/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToPut(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
    Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
```

```
        HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
        response.responseBody().ifPresentOrElse(
            abortableInputStream -> {
                try {
                    IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            },
            () -> logger.error("No response body."));

        logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
    }
} catch (URISyntaxException | IOException e) {
    logger.error(e.getMessage(), e);
}
return byteArrayOutputStream.toByteArray();
}
```

GitHub で [完全な例](#) と [テスト](#) をご覧ください。

アップロード用の署名付き URL を生成して、ファイルをアップロードします (PUT リクエスト)。

以下の例は次の 2 つの部分で構成されます。

- パート 1: Alice はオブジェクトをアップロードするために署名付き URL を生成します。
- パート 2: Bob は署名済み URL を使用してファイルをアップロードします。

パート 1: URL を生成する

Alice はすでに S3 バケットを持っています。彼女は次のコードを使用して、Bob が後続の PUT リクエストで使用できる URL 文字列を生成します。

インポート

```
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();
}
}
```

## パート 2: ファイルオブジェクトをアップロードする

ボブは次の 3 つのコードオプションのいずれかを使用してファイルをアップロードします。

### JDK `URLConnection` を使用する (v1.1 以降)

```
/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
public void useURLConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" + k,
v));

        connection.setRequestMethod("PUT");
        OutputStream out = connection.getOutputStream();

        try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
            FileChannel inChannel = file.getChannel()) {
            ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

            while (inChannel.read(buffer) > 0) {
                buffer.flip();
            }
        }
    }
}
```



```
        for (int i = 0; i < buffer.limit(); i++) {
            out.write(buffer.get());
        }
        buffer.clear();
    }
} catch (IOException e) {
    logger.error(e.getMessage(), e);
}

out.close();
connection.getResponseCode();
logger.info("HTTP response code is " + connection.getResponseCode());

} catch (S3Exception | IOException e) {
    logger.error(e.getMessage(), e);
}
}
```

## JDK `HttpClient` を使用する (v11 以降)

```
/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        final HttpResponse<Void> response = httpClient.send(requestBuilder
            .uri(new URL(presignedUrlString).toURI())
            .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
            .build(),
            HttpResponse.BodyHandlers.discarding());

        logger.info("HTTP response code is " + response.statusCode());

    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}
```

## SDK for Java から `SdkHttpClient` を使用する

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

GitHub で [完全な例](#) と [テスト](#) をご覧ください。

## Amazon S3 のクロスリージョンアクセス

Amazon Simple Storage Service (Amazon S3) バケットで作業をする場合、通常そのバケットの AWS リージョン を知っています。使用するリージョンは、S3 クライアントの作成時に決定されます。

ただし、特定のバケットを操作する必要があったとしても、そのバケットが S3 クライアントに設定されているのと同じリージョンにあるかどうか分からない場合があります。

バケットのリージョンを特定するためにさらに呼び出しを行う代わりに、SDK を使用してさまざまなリージョンの S3 バケットにアクセスできるようにすることができます。

## 設定

クロスリージョンアクセスに関するサポートは、SDK のバージョン 2.20.111 で利用可能になりました。次のスニペットに示すように、Maven ビルドファイルの s3 依存関係には、このバージョンまたはそれ以降のバージョンを使用します。

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.20.111</version>
</dependency>
```

次に、S3 クライアントを作成するときに、スニペットに示すようにクロスリージョンアクセスを有効にします。デフォルトでは、アクセスは有効になっていません。

```
S3AsyncClient client = S3AsyncClient.builder()
    .crossRegionAccessEnabled(true)
    .build();
```

## SDK がクロスリージョンアクセスを提供する方法

putObject メソッドを使用する場合など、リクエストで既存のバケットを参照すると、SDK はクライアント用に設定されたリージョンへのリクエストを開始します。

その特定のリージョンにバケットが存在しない場合、エラーレスポンスにはバケットが存在する実際のリージョンが含まれます。その後、SDK は 2 回目のリクエストで正しいリージョンを使用します。

同じバケットへの今後のリクエストを最適化するために、SDK はこのリージョンマッピングをクライアントにキャッシュします。

## 考慮事項

クロスリージョンバケットアクセスを有効にする場合、バケットがクライアントの設定済みリージョンにない場合、最初の API コールでレイテンシーが増加する可能性があることに注意してください

い。ただし、それ以降のコールではキャッシュされたリージョン情報が有効になるため、パフォーマンスが向上します。

クロスリージョンアクセスを有効にする場合、バケットへのアクセスには影響しません。ユーザーは、そのバケットが存在するリージョンに関係なく、バケットにアクセスする権限を持っている必要があります。

## による Amazon S3 チェックサム

Amazon Simple Storage Service (Amazon S3) では、オブジェクトをアップロードするときにチェックサムを指定できます。チェックサムを指定すると、そのチェックサムはオブジェクトとともに保存され、オブジェクトのダウンロード時に検証できます。

チェックサムは、ファイルを転送する際のデータの整合性をさらに強化します。チェックサムを使用すると、受信したファイルが元のファイルと一致することを確認することで、データ整合性を検証できます。Amazon S3 のチェックサムに関する詳細は、「[Amazon Simple Storage Service ユーザーガイド](#)」を参照してください。

Amazon S3 は現在、SHA-1、SHA-256、CRC-32、CRC-32C の 4 つのチェックサムアルゴリズムをサポートしています。ニーズに最適なアルゴリズムを柔軟に選択して、SDK にチェックサムを計算させることができます。または、サポートされている 4 つのアルゴリズムのいずれかを使用して、事前に計算された独自のチェックサム値を指定することもできます。

チェックサムについては、オブジェクトのアップロードとオブジェクトのダウンロードという 2 つのリクエストフェーズで説明します。

### オブジェクトのアップロード

このアルゴリズムの有効な値はCRC32、CRC32C、SHA1、および SHA256 です。

次のコードスニペットは、CRC-32 チェックサムを含むオブジェクトをアップロードするリクエストを示しています。SDK はリクエストを送信すると、CRC-32 チェックサムを計算してオブジェクトをアップロードします。Amazon S3 はオブジェクトと共にチェックサムを保存します。

SDK が計算するチェックサムが、Amazon S3 がリクエストを受信したときに計算するチェックサムと一致しない場合、エラーが返されます。

事前に計算されたチェックサム値を使用してください。

リクエストで事前に計算されたチェックサム値を指定すると、SDK による自動計算が無効になり、代わりに提供された値が使用されます。

次の例は、事前に計算された SHA-256 チェックサムを含むリクエストを示しています。

Amazon S3 が、指定されたアルゴリズムのチェックサム値が正しくないと判断した場合、サービスはエラーレスポンスを返します。

## マルチパートアップロード

チェックサムはマルチパートアップロードでも使用できます。

## オブジェクトのダウンロード

[getObject](#) メソッドを使用してオブジェクトをダウンロードすると、SDK は

次のスニペット内のリクエストは、チェックサムを計算して値を比較することでレスポンス内のチェックサムを検証するよう SDK に指示します。

オブジェクトがチェックサム付きでアップロードされなかった場合、検証は行われません。

Amazon S3 のオブジェクトには複数のチェックサムを設定できますが、ダウンロード時に検証されるチェックサムは 1 つだけです。SDK が検証するチェックサムは、チェックサムアルゴリズムの効率性に基いた次の優先順位によって決まります。

1. CRC-32C
2. CRC-32
3. SHA-1
4. SHA-256

たとえば、レスポンスに CRC-32 と SHA-256 の両方のチェックサムが含まれている場合、CRC-32 チェックサムのみが検証されます。

## パフォーマンスの高い S3 クライアントを使用する: AWS CRT ベースの S3 クライアント

[AWS 共通ランタイム \(CRT\)](#) 上に構築された AWS CRT ベースの S3 クライアントは、代替の S3 非同期クライアントです。Amazon S3 の [マルチパートアップロード API](#) と [バイト範囲フェッチ](#) を自動的に使用することで、Amazon Simple Storage Service (Amazon S3) との間でオブジェクトを転送し、パフォーマンスと信頼性を向上させます。

AWS CRT ベースの S3 クライアントは、ネットワーク障害が発生した場合の転送の信頼性を向上させます。ファイル転送を最初からやり直すことなく、失敗した個別の部分を再試行することで、信頼性を改善しています。

さらに、AWS CRT ベースの S3 クライアントでは、接続プーリングとドメインネームシステム (DNS) の負荷分散が強化され、スループットも向上します。

SDK の標準 S3 非同期クライアントの代わりに AWS CRT ベースの S3 クライアントを使用すれば、その向上したスループットをすぐに活用できます。

## SDK の AWS CRT ベースのコンポーネント

このトピックで説明する AWS CRT ベースの S3 クライアントと AWS CRT ベースの HTTP クライアントは SDK の異なるコンポーネントです。

AWS CRT ベースの S3 クライアントは [S3AsyncClient](#) インターフェイスの実装であり、Amazon S3 サービスを利用するために使用されます。これは S3AsyncClient インターフェイスの Java ベースの実装に代わるもので、いくつかの利点があります。

[AWS CRT ベースの HTTP クライアント](#)は、[SDKAsyncHttpClient](#) インターフェイスを実装したもので、一般的な HTTP 通信に使用されます。これは SdkAsyncHttpClient インターフェイスの Netty 実装に代わるもので、いくつかの利点があります。

どちらのコンポーネントも [AWS 共通ランタイム](#)のライブラリを使用していますが、AWS CRT ベースの S3 クライアントは [aws-c-s3 ライブラリ](#)を使用し、[S3 マルチパートアップロード API](#) 機能をサポートしています。AWS CRT ベースの HTTP クライアントは汎用的な使用を目的としているため、S3 マルチパートアップロード API 機能はサポートしていません。

## AWS CRT ベースの S3 クライアントを使用する依存関係を追加

AWS CRT ベースの S3 クライアントを使用するには、次の 2 つの依存関係を Maven プロジェクトファイルに追加します。この例は、使用する最小バージョンを示しています。Maven central リポジトリで [s3](#) と [aws-crt](#) のアーティファクトの最新バージョンを検索します。

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.20.68</version>
</dependency>
<dependency>
  <groupId>software.amazon.awssdk.crt</groupId>
```

```
<artifactId>aws-crt</artifactId>
<version>0.21.16</version>
</dependency>
```

AWS CRT ベースの S3 クライアントのインスタンスを作成します。

以下のコードスニペットに示すように、デフォルト設定で AWS CRT ベースの S3 クライアントのインスタンスを作成します。

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
```

クライアントを設定するには、AWS CRT クライアントビルダーを使用します。ビルダーの方法を変更することで、標準の S3 非同期クライアントから AWS CRT ベースのクライアントに切り替えることができます。

```
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

スタンダードビルダーの設定の一部は、現在 AWS CRT クライアントビルダーではサポートされていない場合があります。スタンダードビルダーを取得するには、`S3AsyncClient#builder()` を呼び出します。

## AWS CRT ベースの S3 クライアントを使用する

AWS CRT ベースの S3 クライアントを使用して Amazon S3 API オペレーションを呼び出します。次の例は、AWS SDK for Java を介して実行できる [PutObject](#) オペレーションと [GetObject](#) オペレーションを示しています。

```
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>),
        AsyncResponseTransformerToFile(Paths.get(<FILE_NAME>)))
        .join();
```

## Amazon S3 Transfer Manager を使用してファイルとディレクトリを転送する

Amazon S3 転送マネージャは、AWS SDK for Java 2.x用のオープンソースの高レベルファイル転送ユーティリティです。Amazon Simple Storage Service (Amazon S3) との間でファイルやディレクトリを転送するために使用します。

[AWS CRT ベースの S3 クライアント](#) 上に構築すると、S3 Transfer Manager では[マルチパートアップロード API](#) や[バイト範囲フェッチ](#)などのパフォーマンスの向上を活用できます。

S3 Transfer Manager では、転送の進行状況をリアルタイムでモニタリングし、転送を一時停止して後で実行することもできます。



## 使用を開始する

### ビルドファイルに依存関係を追加する

AWS CRT ベースの S3 クライアントに基づく拡張パフォーマンスで S3 Transfer Manager を使用するには、次の依存関係を使用してビルドファイルを設定します。

- SDK for Java 2.x のバージョン **2.19.1** 以降を使用します。
- `s3-transfer-manager` アーティファクトを依存関係として追加します。
- バージョン **0.20.3** 以降の `aws-crt` アーティファクトを依存関係として追加します。

次のコード例は、プロジェクトの Maven 依存関係を設定する方法を示しています。

```
<project>
  <properties>
    <aws.sdk.version>2.19.1</aws.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>s3-transfer-manager</artifactId>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk.crt</groupId>
      <artifactId>aws-crt</artifactId>
      <version>0.20.3</version>
    </dependency>
  </dependencies>
</project>
```

Maven central リポジトリで [s3-transfer-manager](#) と [aws-crt](#) のアーティファクトの最新バージョンを検索します。

## S3 Transfer Manager のインスタンスを作成する

次のスニペットは、デフォルト設定で [S3TransferManager](#) インスタンスを作成する方法を示しています。

```
S3TransferManager transferManager = S3TransferManager.create();
```

次の例は、カスタム設定で S3 Transfer Manager を設定する方法を示しています。この例では、[AWS CRT ベースの S3AsyncClient](#) インスタンスが S3 Transfer Manager の基盤となるクライアントとして使用されます。

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .targetThroughputInGbps(20.0)
    .minimumPartSizeInBytes(8 * MB)
    .build();

S3TransferManager transferManager = S3TransferManager.builder()
    .s3Client(s3AsyncClient)
    .build();
```

### Note

`aws-crt` 依存関係がビルドファイルに含まれていない場合、S3 Transfer Manager は SDK for Java 2.x で使用されている標準 S3 非同期クライアント上に構築されます。

## S3 バケットにファイルをアップロードする

次の例は、ファイルのアップロード例と [LoggingTransferListener](#)、アップロードの進行状況をログに記録する のオプションの使用を示しています。

S3 Transfer Manager を使用して Amazon S3 にファイルをアップロードするには、`S3TransferManager` の [uploadFile](#) メソッドに [UploadFileRequest](#) オブジェクトを渡します。

`uploadFile` メソッドから返される [FileUpload](#) オブジェクトは、アップロードプロセスを表します。リクエストが完了すると、[CompletedFileUpload](#) オブジェクトにはアップロードに関する情報が含まれます。

```
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();
}
```

## インポート

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;
```

## S3 バケットからのファイルのダウンロード

次の例は、ダウンロードの進行状況をログ [LoggingTransferListener](#) に記録する をオプションで使用してダウンロード例を示しています。

S3 Transfer Manager を使用して S3 バケットからオブジェクトをダウンロードするには、[DownloadFileRequest](#) オブジェクトをビルドして [downloadFile](#) メソッドに渡します。

S3TransferManagerの `downloadFile` メソッドによって返される [FileDownload](#) オブジェクトは、ファイル転送を表します。ダウンロードが完了すると、にはダウンロードに関する情報へのアクセス [CompletedFileDownload](#) が含まれます。

```
public Long downloadFile(S3TransferManager transferManager, String bucketName,
                        String key, String downloadedFilePath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .destination(Paths.get(downloadedFilePath))
        .build();

    FileDownload downloadFile = transferManager.downloadFile(downloadFileRequest);

    CompletedFileDownload downloadResult = downloadFile.completionFuture().join();
    logger.info("Content length [{}]", downloadResult.response().contentLength());
    return downloadResult.response().contentLength();
}
```

## インポート

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;
```

## Amazon S3 バケットを他のバケットにコピーする

次の例は、S3 Transfer Manager を使用してオブジェクトをコピーする方法を示しています。

S3 バケットから別のバケットへのオブジェクトのコピーを開始するには、基本 [CopyObjectRequest](#) インスタンスを作成します。

次に、S3 Transfer Manager [CopyRequest](#) が使用できる `CopyObjectRequest` で基本 をラップします。

`S3TransferManager` の `copy` メソッドによって返される `Copy` オブジェクトはコピープロセスを表します。コピープロセスが完了すると、[CompletedCopy](#) オブジェクトにはレスポンスに関する詳細が含まれます。

```
public String copyObject(S3TransferManager transferManager, String bucketName,
    String key, String destinationBucket, String destinationKey) {
    CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
        .sourceBucket(bucketName)
        .sourceKey(key)
        .destinationBucket(destinationBucket)
        .destinationKey(destinationKey)
        .build();

    CopyRequest copyRequest = CopyRequest.builder()
        .copyObjectRequest(copyObjectRequest)
        .build();

    Copy copy = transferManager.copy(copyRequest);

    CompletedCopy completedCopy = copy.completionFuture().join();
    return completedCopy.response().copyObjectResult().eTag();
}
```

### Note

S3 Transfer Manager を使用してクロスリージョンコピーを実行するには、次のスニペットに示すように、AWS CRT ベースの S3 クライアントビルダー `crossRegionAccessEnabled` を有効にします。

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
    .crossRegionAccessEnabled(true)
    .build();

S3TransferManager transferManager = S3TransferManager.builder()
    .s3Client(s3AsyncClient)
```

```
.build();
```

## インポート

```
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;
```

## ローカルディレクトリを S3 バケットにアップロードする

次の例は、ローカルディレクトリを S3 にアップロードする方法を示しています。

まず、S3TransferManager インスタンスの [uploadDirectory](#) メソッドを呼び出し、`UploadDirectoryRequest` を渡します。

[DirectoryUpload](#) オブジェクトはアップロードプロセスを表し、リクエストが完了すると `CompletedDirectoryUpload` オブジェクトが生成されます。CompleteDirectoryUpload オブジェクトは、どのファイルが転送に失敗したかなど、転送の結果に関する情報を含めます。

```
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();
}
```

```
}
```

## インポート

```
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;
```

## ローカルディレクトリへの S3 バケットオブジェクトのダウンロード

次の例に示すように、S3 バケットのオブジェクトをローカルディレクトリにダウンロードできません。

S3 バケット内のオブジェクトをローカルディレクトリにダウンロードするには、まず Transfer Manager の [downloadDirectory](#) メソッドを呼び出し、を渡します [DownloadDirectoryRequest](#)。

[DirectoryDownload](#) オブジェクトはダウンロードプロセスを表し、リクエストが完了すると [CompleteDirectoryDownload](#) オブジェクトは、どのファイルが転送に失敗したかなど、転送の結果に関する情報を含めます。

```
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();
}
```

## インポート

```
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;
```

## 完全な例を見る

には [GitHub](#)、[このページのすべての例の完全なコードが含まれています。](#)

# Amazon Simple Notification Service を使用する

Amazon Simple Notification Service では、アプリケーションからのリアルタイムの通知メッセージを複数の通信チャネル経由でサブスクライバーに簡単にプッシュできます。このトピックでは、Amazon SNS の基本機能の一部を実行する方法について説明します。

## トピックの作成

トピックは、メッセージの送信先システムを定義する通信チャネルの論理的なグループです。たとえば、AWS Lambda や HTTP ウェブフックへのメッセージのファンアウトなどです。Amazon SNS に



メッセージを送信すると、トピックで定義されているチャンネルに配信されます。これにより、サブスクライバーがメッセージを使用できるようになります。

トピックを作成するには、最初に、ビルダーの `name()` メソッドを使用して設定されたトピックの名前で [CreateTopicRequest](#) オブジェクトを構築します。その後、[SnsClient](#) の `createTopic()` メソッドを使用して、リクエストオブジェクトを Amazon SNS に送信します。次のコードスニペットに示すように、このリクエストの結果を [CreateTopicResponse](#) オブジェクトとしてキャプチャできます。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

## Code

```
public static String createSNSTopic(SnsClient snsClient, String topicName ) {

    CreateTopicResponse result = null;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();
    } catch (SnsException e) {

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

[GitHub](#) で完全な例をご覧ください。

## Amazon SNS トピックをリストする

既存の Amazon SNS トピックのリストを取得するには、[ListTopicsRequest](#) オブジェクトを構築します。次に、SnsClient の listTopics() メソッドを使用して、リクエストオブジェクトを Amazon SNS に送信します。このリクエストの結果は、[ListTopicsResponse](#) オブジェクトとしてキャプチャできます。

次のコードスニペットは、リクエストの HTTP ステータスコードと、Amazon SNS トピックの Amazon リソースネーム (ARN) のリストを出力します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

### Code

```
public static void listSNSTopics(SnsClient snsClient) {

    try {
        ListTopicsRequest request = ListTopicsRequest.builder()
            .build();

        ListTopicsResponse result = snsClient.listTopics(request);
        System.out.println("Status was " + result.sdkHttpResponse().statusCode() +
            "\n\nTopics\n\n" + result.topics());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

[GitHub](#) で完全な例をご覧ください。

## トピックへのエンドポイントのサブスクライブ

トピックを作成したら、そのトピックのエンドポイントとなる通信チャネルを設定できます。メッセージは、Amazon SNS による受信後にこれらのエンドポイントに配信されます。

通信チャンネルをトピックのエンドポイントとして設定するには、そのエンドポイントをトピックにサブスクライブします。開始するには、[SubscribeRequest](#) オブジェクトを構築します。通信チャンネル (lambda や email など) を `protocol()` として指定します。 `endpoint()` を関連する出力場所 (Lambda 関数の ARN や E メールアドレスなど) に設定し、サブスクライブするトピックの ARN を `topicArn()` として設定します。 `SnsClient` の `subscribe()` メソッドを使用して、リクエストオブジェクトを Amazon SNS に送信します。このリクエストの結果は、[SubscribeResponse](#) オブジェクトとしてキャプチャできます。

次のコードスニペットは、トピックに E メールアドレスをサブスクライブする方法を示しています。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
```

## Code

```
public static void subEmail(SnsClient snsClient, String topicArn, String email) {

    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("email")
            .endpoint(email)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n\n"
            + "Status is " + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

[GitHub](#) で完全な例をご覧ください。

## トピックへのメッセージの発行

トピックと 1 つ以上のエンドポイントを設定したら、そのトピックにメッセージを発行できます。開始するには、[PublishRequest](#) オブジェクトを構築します。送信する `message()` と、送信先のトピック (`topicArn()`) の ARN を指定します。次に、`SnsClient` の `publish()` メソッドを使用して、リクエストオブジェクトを Amazon SNS に送信します。このリクエストの結果は、[PublishResponse](#) オブジェクトとして取得できます。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

### Code

```
public static void pubTopic(SnsClient snsClient, String message, String topicArn) {

    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .topicArn(topicArn)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out.println(result.messageId() + " Message sent. Status is " +
            result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

[GitHub](#) で完全な例をご覧ください。

## トピックからのエンドポイントの登録解除

トピックのエンドポイントとして設定されている通信チャンネルを削除できます。その後、トピック自体は引き続き存在し、そのトピックに設定されている他のエンドポイントにメッセージを配信します。

トピックのエンドポイントとしての通信チャンネルを削除するには、そのエンドポイントをトピックから登録解除します。開始するには、[UnsubscribeRequest](#) オブジェクトを構築し、登録解除するトピックの ARN を `subscriptionArn()` として設定します。次に、`SnsClient` の `unsubscribe()` メソッドを使用して、リクエストオブジェクトを SNS に送信します。このリクエストの結果は、[UnsubscribeResponse](#) オブジェクトとしてキャプチャできます。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
```

### Code

```
public static void unSub(SnsClient snsClient, String subscriptionArn) {

    try {
        UnsubscribeRequest request = UnsubscribeRequest.builder()
            .subscriptionArn(subscriptionArn)
            .build();

        UnsubscribeResponse result = snsClient.unsubscribe(request);

        System.out.println("\n\nStatus was " +
            result.sdkHttpResponse().statusCode()
            + "\n\nSubscription was removed for " + request.subscriptionArn());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

[GitHub](#) で完全な例をご覧ください。

## トピックの削除

Amazon SNS トピックを削除するには、最初に、トピックの ARN をビルダーの `topicArn()` メソッドとして設定して [DeleteTopicRequest](#) オブジェクトを構築します。次に、`SnsClient` の `deleteTopic()` メソッドを使用して、リクエストオブジェクトを Amazon SNS に送信します。次のコードスニペットに示すように、このリクエストの結果を [DeleteTopicResponse](#) オブジェクトとしてキャプチャできます。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

### Code

```
public static void deleteSNSTopic(SnsClient snsClient, String topicArn ) {

    try {
        DeleteTopicRequest request = DeleteTopicRequest.builder()
            .topicArn(topicArn)
            .build();

        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

[GitHub](#) で完全な例をご覧ください。

詳細については、「Amazon Simple Notification Service デベロッパーガイド」を参照してください。

## の操作 Amazon Simple Queue Service

このセクションでは、AWS SDK for Java 2.x [Amazon Simple Queue Service](#)を使用したプログラミングの例を示します。

次の例には各手法を示すのに必要なコードのみが含まれます。[完全なサンプルコードは、「」で入手できます GitHub](#)。そこから、単一のソースファイルをダウンロードするかリポジトリをローカルにクローン作成して、ビルドし実行するためのすべての例を取得できます。

### トピック

- [Amazon Simple Queue Service メッセージキューを使用する](#)
- [Amazon Simple Queue Service メッセージを送信、受信、および削除する](#)

## Amazon Simple Queue Service メッセージキューを使用する

メッセージキューは、Amazon Simple Queue Service でメッセージを確実に送信するために使用する論理コンテナです。キューには、標準と先入れ先出し (FIFO) の 2 種類があります。キューおよびキュータイプ間の相違点の詳細については、「[Amazon Simple Queue Service デベロッパーガイド](#)」を参照してください。

このトピックでは、AWS SDK for Java を使用して Amazon Simple Queue Service キューの URL の作成、一覧表示、削除、および取得を行う方法について説明します。

次の例で使用されている `sqsClient` 変数は、次のスニペットから作成できます。

```
SqsClient sqsClient = SqsClient.create();
```

静的 `create()` メソッド `SqsClient` を使用してを作成すると、SDK は [デフォルトのリージョンプロバイダーチェーン](#) を使用してリージョンを設定し、[デフォルトの認証情報プロバイダーチェーン](#) を使用して認証情報を設定します。

### キューを作成する

次のコードスニペットに示すように、`SqsClient` の `sqsClient.createQueue` メソッドを使用して、キューパラメータを記述する [CreateQueueRequest](#) オブジェクトを指定します。

### インポート

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Code

```
CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
    .queueName(queueName)
    .build();

sqsClient.createQueue(createQueueRequest);
```

「」で[完全なサンプル](#)を参照してください GitHub。

## キューの一覧表示

アカウントのAmazon Simple Queue Serviceキューを一覧表示するには、[ListQueuesRequest](#) オブジェクトを使用して SqsClient の `listQueues` メソッドを呼び出します。

パラメータを受け取らない [listQueues](#) メソッドの形式を使用すると、サービスはすべてのキューを返します。最大 1,000 キューです。

オブジェクトにキュー名のプレフィックスを指定 [ListQueuesRequest](#) して、次のコードに示すように、そのプレフィックスに一致するキューに結果を制限できます。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Code

```
String prefix = "que";

try {
    ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
    ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
```



```
        for (String url : listQueuesResponse.queueUrls()) {
            System.out.println(url);
        }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

「」で[完全なサンプル](#)を参照してください GitHub。

## キューの URL の取得

次のコードは、[GetQueueUrlRequest](#) オブジェクトで `SqsClient` の `getQueueUrl` メソッドを呼び出してキューの URL を取得する方法を示しています。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

### Code

```
GetQueueUrlResponse getQueueUrlResponse =

sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
String queueUrl = getQueueUrlResponse.queueUrl();
return queueUrl;
```

「」で[完全なサンプル](#)を参照してください GitHub。

## キューの削除

[DeleteQueueRequest](#) オブジェクトへのキューの [URL](#) を指定します。次に、次のコードに示すように、`SqsClient` の `deleteQueue` メソッドを呼び出してキューを削除します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
```

```
import java.util.List;
```

## Code

```
public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {  
  
    try {  
  
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()  
            .queueName(queueName)  
            .build();  
  
        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();  
  
        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()  
            .queueUrl(queueUrl)  
            .build();  
  
        sqsClient.deleteQueue(deleteQueueRequest);  
  
    } catch (SqsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

「」で[完全なサンプル](#)を参照してください GitHub。

## 詳細情報

- [CreateQueue](#) Amazon Simple Queue Service API リファレンスの
- [GetQueueUrl](#) Amazon Simple Queue Service API リファレンスの
- [ListQueues](#) Amazon Simple Queue Service API リファレンスの
- [DeleteQueue](#) Amazon Simple Queue Service API リファレンスの

## Amazon Simple Queue Service メッセージを送信、受信、および削除する

メッセージは、分散コンポーネントによって送受信できるデータの一部です。メッセージは、常に [SQS キュー](#) を使用して提供されます。

次の例で使用されている `sqsClient` 変数は、次のスニペットから作成できます。

```
SqsClient sqsClient = SqsClient.create();
```

静的 `create()` メソッド `SqsClient` を使用してを作成すると、SDK は [デフォルトのリージョンプロバイダーチェーン](#) を使用してリージョンを設定し、[デフォルトの認証情報プロバイダーチェーン](#) を使用して認証情報を設定します。

## メッセージの送信

`SqsClient` クライアント `sendMessage` メソッドを呼び出して Amazon Simple Queue Service キューに 1 つのメッセージを追加します。キューの [URL](#)、メッセージ本文、およびオプションの遅延値 (秒単位) を含む [SendMessageRequest](#) オブジェクトを指定します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

### Code

```
sqsClient.sendMessage(SendMessageRequest.builder()
    .queueUrl(queueUrl)
    .messageBody("Hello world!")
    .delaySeconds(10)
    .build());

sqsClient.sendMessage(sendMsgRequest);
```

## リクエストで複数のメッセージを送信する

`SqsClient` の `sendMessageBatch` メソッドを使用して 1 つのリクエストで複数のメッセージを送信します。このメソッドは、キュー URL と送信 [SendMessageBatchRequest](#) するメッセージのリストを含むを受け取ります。(各メッセージは [SendMessageBatchRequestEntry](#) です。) また、メッセージの遅延値を設定して、特定のメッセージの送信を遅延させることもできます。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
```

```
import java.util.List;
```

## Code

```
SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
    .queueUrl(queueUrl)

    .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from msg
1").build()),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10).build())
    .build();
sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

「」で[完全なサンプル](#)を参照してください GitHub。

## メッセージを取得する

キューに現在含まれているメッセージを取得するには、SqsClient の `receiveMessage` メソッドを呼び出します。このメソッドは、キュー URL [ReceiveMessageRequest](#) を含む `ReceiveMessageRequest` を受け取ります。また、返るメッセージの最大数を指定することもできます。メッセージは、[Message](#) オブジェクトのリストとして返されます。

## インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Code

```
try {
    ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
    .queueUrl(queueUrl)
    .numberOfMessages(5)
    .build();
    List<Message> messages =
sqsClient.receiveMessage(receiveMessageRequest).messages();
```

```
        return messages;
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
```

「」で[完全なサンプル](#)を参照してください GitHub。

## 受信後にメッセージを削除する

メッセージを受信し、その内容を処理した後、メッセージの受信ハンドルとキュー URL を `SqsClient.deleteMessage` メソッドに送信して、キューからメッセージを削除します。

### インポート

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

### Code

```
try {
    for (Message message : messages) {
        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
        .queueUrl(queueUrl)
        .receiptHandle(message.receiptHandle())
        .build();
        sqsClient.deleteMessage(deleteMessageRequest);
    }
}
```

「」で[完全なサンプル](#)を参照してください GitHub。

## 詳細情報

- Amazon Simple Queue Service デベロッパーガイドの [Amazon Simple Queue Service キューの仕組み](#)
- [SendMessage](#) Amazon Simple Queue Service API リファレンスの
- [SendMessageBatch](#) Amazon Simple Queue Service API リファレンスの

- [ReceiveMessage](#) Amazon Simple Queue Service API リファレンスの
- [DeleteMessage](#) Amazon Simple Queue Service API リファレンスの

## Amazon Transcribe を使用する

以下の例では、Amazon Transcribe を使用した双方向ストリーミングの仕組みを示しています。双方向ストリーミングとは、サービスに送信されるデータストリームと、サービスからリアルタイムで受信されるデータストリームが、両方あることを意味します。この例では、Amazon Transcribe の文字起こしストリーミングを使用して、音声ストリームを送信し、文字起こしされたテキストストリームをリアルタイムで受信します。

この機能の詳細については、Amazon Transcribe デベロッパーガイドの[ストリーミング文字起こし](#)を参照してください。

Amazon Transcribe の使用を開始するには、Amazon Transcribe デベロッパーガイドの[開始方法](#)を参照してください。

## マイクを設定する

このコードは、`javax.sound.sampled` パッケージを使用して、入力デバイスからの音声をストリーミングします。

### Code

```
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.TargetDataLine;

public class Microphone {

    public static TargetDataLine get() throws Exception {
        AudioFormat format = new AudioFormat(16000, 16, 1, true, false);
        DataLine.Info datalineInfo = new DataLine.Info(TargetDataLine.class, format);

        TargetDataLine dataLine = (TargetDataLine) AudioSystem.getLine(datalineInfo);
        dataLine.open(format);

        return dataLine;
    }
}
```

```
}
```

[GitHub](#) で完全な例をご覧ください。

## パブリッシャーを作成する

このコードは、Amazon Transcribe 音声ストリームから音声データを発行するパブリッシャーを実装します。

### Code

```
package com.amazonaws.transcribe;

import java.io.IOException;
import java.io.InputStream;
import java.io.UncheckedIOException;
import java.nio.ByteBuffer;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.atomic.AtomicLong;
import org.reactivestreams.Publisher;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.transcribestreaming.model.AudioEvent;
import software.amazon.awssdk.services.transcribestreaming.model.AudioStream;
import
    software.amazon.awssdk.services.transcribestreaming.model.TranscribeStreamingException;

public class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;

    public AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {
        s.onSubscribe(new SubscriptionImpl(s, inputStream));
    }

    private class SubscriptionImpl implements Subscription {
```

```
private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
private ExecutorService executor = Executors.newFixedThreadPool(1);
private AtomicLong demand = new AtomicLong(0);

private final Subscriber<? super AudioStream> subscriber;
private final InputStream inputStream;

private SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
inputStream) {
    this.subscriber = s;
    this.inputStream = inputStream;
}

@Override
public void request(long n) {
    if (n <= 0) {
        subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
    }

    demand.getAndAdd(n);

    executor.submit(() -> {
        try {
            do {
                ByteBuffer audioBuffer = getNextEvent();
                if (audioBuffer.remaining() > 0) {
                    AudioEvent audioEvent = audioEventFromBuffer(audioBuffer);
                    subscriber.onNext(audioEvent);
                } else {
                    subscriber.onComplete();
                    break;
                }
            } while (demand.decrementAndGet() > 0);
        } catch (TranscribeStreamingException e) {
            subscriber.onError(e);
        }
    });
}

@Override
public void cancel() {
}
```



```
private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

    int len = 0;
    try {
        len = inputStream.read(audioBytes);

        if (len <= 0) {
            audioBuffer = ByteBuffer.allocate(0);
        } else {
            audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
        }
    } catch (IOException e) {
        throw new UncheckedIOException(e);
    }

    return audioBuffer;
}

private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
```

[GitHub](#) で完全な例をご覧ください。

## クライアントを作成してストリーミングを開始する

main メソッドで、リクエストオブジェクトを作成し、音声入カストリームを開始して、音声入力パブリッシャーをインスタンス化します。

[StartStreamTranscriptionResponseHandler](#) を作成して、Amazon Transcribe からのレスポンスを処理する方法を指定する必要があります。

その後、TranscribeStreamingAsyncClient の startStreamTranscription メソッドを使用して双方向ストリーミングを開始します。

### インポート

```
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.TargetDataLine;
import javax.sound.sampled.AudioInputStream;
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.transcribestreaming.TranscribeStreamingAsyncClient;
import
    software.amazon.awssdk.services.transcribestreaming.model.TranscribeStreamingException ;
import
    software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionRequest;
import software.amazon.awssdk.services.transcribestreaming.model.MediaEncoding;
import software.amazon.awssdk.services.transcribestreaming.model.LanguageCode;
import
    software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionResponseHandler;
import software.amazon.awssdk.services.transcribestreaming.model.TranscriptEvent;
```

## Code

```
public static void convertAudio(TranscribeStreamingAsyncClient client) throws
Exception {

    try {

        StartStreamTranscriptionRequest request =
StartStreamTranscriptionRequest.builder()
            .mediaEncoding(MediaEncoding.PCM)
            .languageCode(LanguageCode.EN_US)
            .mediaSampleRateHertz(16_000).build();

        TargetDataLine mic = Microphone.get();
        mic.start();

        AudioStreamPublisher publisher = new AudioStreamPublisher(new
AudioInputStream(mic));

        StartStreamTranscriptionResponseHandler response =
            StartStreamTranscriptionResponseHandler.builder().subscriber(e -> {
                TranscriptEvent event = (TranscriptEvent) e;
                event.transcript().results().forEach(r ->
r.alternatives().forEach(a -> System.out.println(a.transcript())));
            }).build();
```

```
        // Keeps Streaming until you end the Java program
        client.startStreamTranscription(request, publisher, response);

    } catch (TranscribeStreamingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

[GitHub](#) で完全な例をご覧ください。

## 詳細情報

- Amazon Transcribe デベロッパーガイドの[仕組み](#)。
- Amazon Transcribe デベロッパーガイドの[ストリーミングオーディオの開始方法](#)。

## SDK for Java 2.x のコード例

このトピックのコード例は、AWS SDK for Java 2.x で を使用する方法を示しています AWS。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

クロスサービスの例は、複数の AWS のサービスで動作するサンプルアプリケーションです。

例

- [SDK for Java 2.x を使用するアクションとシナリオ](#)
- [SDK for Java 2.x を使用したクロスサービスの例](#)

## SDK for Java 2.x を使用するアクションとシナリオ

次のコード例は、AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS のサービス。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

サービス

- [SDK for Java 2.x を使用した API Gateway の例](#)
- [SDK for Java 2.x を使用した Application Auto Scaling の例](#)
- [SDK for Java 2.x を使用した Application Recovery Controller の例](#)
- [SDK for Java 2.x を使用した Aurora の例](#)
- [SDK for Java 2.x を使用した Auto Scaling の例](#)

- [SDK for Java 2.x を使用する Amazon Bedrock の例](#)
- [SDK for Java 2.x を使用する Amazon Bedrock ランタイムの例](#)
- [CloudFront SDK for Java 2.x を使用した の例](#)
- [CloudWatch SDK for Java 2.x を使用した の例](#)
- [CloudWatch SDK for Java 2.x を使用したイベントの例](#)
- [CloudWatch SDK for Java 2.x を使用したログの例](#)
- [SDK for Java 2.x を使用した Amazon Cognito ID の例](#)
- [SDK for Java 2.x を使用した Amazon Cognito ID プロバイダーの例](#)
- [SDK for Java 2.x を使用する Amazon Comprehend の例](#)
- [SDK for Java 2.x を使用した DynamoDB の例](#)
- [SDK for Java 2.x を使用する Amazon EC2 の例](#)
- [SDK for Java 2.x を使用する Amazon ECS の例](#)
- [Elastic Load Balancing - SDK for Java 2.x を使用したバージョン 2 の例](#)
- [MediaStore SDK for Java 2.x を使用した の例](#)
- [OpenSearch SDK for Java 2.x を使用したサービス例](#)
- [EventBridge SDK for Java 2.x を使用した の例](#)
- [SDK for Java 2.x を使用した Forecast の例](#)
- [AWS Glue SDK for Java 2.x を使用した の例](#)
- [HealthImaging SDK for Java 2.x を使用した の例](#)
- [SDK for Java 2.x を使用した IAM の例](#)
- [AWS IoT SDK for Java 2.x を使用した の例](#)
- [AWS IoT data SDK for Java 2.x を使用した の例](#)
- [SDK for Java 2.x を使用する Amazon Keyspaces の例](#)
- [SDK for Java 2.x を使用した Kinesis の例](#)
- [AWS KMS SDK for Java 2.x を使用した の例](#)
- [SDK for Java 2.x を使用した Lambda の例](#)
- [MediaConvert SDK for Java 2.x を使用した の例](#)
- [SDK for Java 2.x を使用した Migration Hub の例](#)

- [SDK for Java 2.x を使用した Amazon Personalize の例](#)
- [SDK for Java 2.x を使用した Amazon Personalize Events の例](#)
- [SDK for Java 2.x による Amazon Personalize Runtime の例](#)
- [SDK for Java 2.x を使用した Amazon Pinpoint の例](#)
- [SDK for Java 2.x を使用した Amazon Pinpoint SMS および音声 API の例](#)
- [SDK for Java 2.x を使用する Amazon Polly の例](#)
- [SDK for Java 2.x を使用する Amazon RDS の例](#)
- [SDK for Java 2.x を使用した Amazon Redshift の例](#)
- [SDK for Java 2.x を使用する Amazon Rekognition の例](#)
- [SDK for Java 2.x を使用した Route 53 ドメイン登録のコード例](#)
- [SDK for Java 2.x を使用する Amazon S3 の例](#)
- [SDK for Java 2.x を使用した S3 Glacier の例](#)
- [SageMaker SDK for Java 2.x を使用した の例](#)
- [SDK for Java 2.x を使用した Secrets Manager の例](#)
- [SDK for Java 2.x を使用する Amazon SES の例](#)
- [SDK for Java 2.x を使用する Amazon SES API v2 の例](#)
- [SDK for Java 2.x を使用する Amazon SNS の例](#)
- [SDK for Java 2.x を使用する Amazon SQS の例](#)
- [SDK for Java 2.x を使用した Step Function の例](#)
- [AWS STS SDK for Java 2.x を使用した の例](#)
- [AWS Support SDK for Java 2.x を使用した の例](#)
- [SDK for Java 2.x を使用した Systems Manager の例](#)
- [SDK for Java 2.x を使用する Amazon Textract の例](#)
- [SDK for Java 2.x を使用した Amazon Transcribe の例](#)

## SDK for Java 2.x を使用した API Gateway の例

次のコード例は、API Gateway AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック


- [アクション](#)

## アクション

### CreateDeployment

次の例は、CreateDeployment を使用する方法を説明しています。

SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
}
```

- API の詳細については、「API リファレンス [CreateDeployment](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateRestApi

次の例は、CreateRestApi を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";  
}
```

- API の詳細については、「API リファレンス [CreateRestApi](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteDeployment

次の例は、DeleteDeployment を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);  
    }  
}
```

- API の詳細については、「API リファレンス [DeleteDeployment](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteRestApi

次の例は、DeleteRestApi を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);  
    }  
}
```

- API の詳細については、「API リファレンス [DeleteRestApi](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## SDK for Java 2.x を使用した Application Auto Scaling の例

次のコード例は、Application Auto Scaling AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- [アクション](#)

## アクション

### DeleteScalingPolicy

次の例は、DeleteScalingPolicy を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingException;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DeregisterScalableTargetRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResponse;
```

```
import
software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import
software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DisableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableId> <policyName>\s

            Where:
                tableId - The table Id value (for example, table/Music).\s
                policyName - The name of the policy (for example, $Music5-scaling-
policy).

            """;
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        ApplicationAutoScalingClient appAutoScalingClient =
ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        String tableId = args[0];
        String policyName = args[1];

        deletePolicy(appAutoScalingClient, policyName, tableWCUs, ns, tableId);
    }
}
```

```
        verifyScalingPolicies(appAutoScalingClient, tableId, ns, tableWCUs);
        deregisterScalableTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
    }

    public static void deletePolicy(ApplicationAutoScalingClient
appAutoScalingClient, String policyName, ScalableDimension tableWCUs,
ServiceNamespace ns, String tableId) {
        try {
            DeleteScalingPolicyRequest delSPRequest =
DeleteScalingPolicyRequest.builder()
                .policyName(policyName)
                .scalableDimension(tableWCUs)
                .serviceNamespace(ns)
                .resourceId(tableId)
                .build();

            appAutoScalingClient.deleteScalingPolicy(delSPRequest);
            System.out.println(policyName + " was deleted successfully.");

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Verify that the scaling policy was deleted
    public static void verifyScalingPolicies(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        DescribeScalingPoliciesRequest dscRequest =
DescribeScalingPoliciesRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceId(tableId)
            .build();

        DescribeScalingPoliciesResponse response =
appAutoScalingClient.describeScalingPolicies(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }
}
```

```
public static void deregisterScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    try {
        DeregisterScalableTargetRequest targetRequest =
DeregisterScalableTargetRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceId(tableId)
            .build();

        appAutoScalingClient.deregisterScalableTarget(targetRequest);
        System.out.println("The scalable target was deregistered.");

    } catch (ApplicationAutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
        .scalableDimension(tableWCUs)
        .serviceNamespace(ns)
        .resourceIds(tableId)
        .build();

    DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(response);
}
}
```

- APIの詳細については、「API リファレンス [DeleteScalingPolicy](#)」の「」を参照してください。AWS SDK for Java 2.x

## RegisterScalableTarget

次の例は、RegisterScalableTarget を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingException;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResponse;
import software.amazon.awssdk.services.applicationautoscaling.model.PolicyType;
import
    software.amazon.awssdk.services.applicationautoscaling.model.PredefinedMetricSpecification;
import
    software.amazon.awssdk.services.applicationautoscaling.model.PutScalingPolicyRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.RegisterScalableTargetRequest;
import software.amazon.awssdk.services.applicationautoscaling.model.ScalingPolicy;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import software.amazon.awssdk.services.applicationautoscaling.model.MetricType;
import
    software.amazon.awssdk.services.applicationautoscaling.model.TargetTrackingScalingPolicyConfiguration;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class EnableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableId> <roleARN> <policyName>\s

            Where:
                tableId - The table Id value (for example, table/Music).
                roleARN - The ARN of the role that has ApplicationAutoScaling
permissions.
                policyName - The name of the policy to create.

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        System.out.println("This example registers an Amazon DynamoDB table, which
is the resource to scale.");
        String tableId = args[0];
        String roleARN = args[1];
        String policyName = args[2];
        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        ApplicationAutoScalingClient appAutoScalingClient =
ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        registerScalableTarget(appAutoScalingClient, tableId, roleARN, ns,
tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        configureScalingPolicy(appAutoScalingClient, tableId, ns, tableWCUs,
policyName);
    }
}
```



```
public static void registerScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, String roleARN, ServiceNamespace ns,
ScalableDimension tableWCUs) {
    try {
        RegisterScalableTargetRequest targetRequest =
RegisterScalableTargetRequest.builder()
            .serviceNamespace(ns)
            .scalableDimension(tableWCUs)
            .resourceId(tableId)
            .roleARN(roleARN)
            .minCapacity(5)
            .maxCapacity(10)
            .build();

        appAutoScalingClient.registerScalableTarget(targetRequest);
        System.out.println("You have registered " + tableId);

    } catch (ApplicationAutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

// Verify that the target was created.
public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
        .scalableDimension(tableWCUs)
        .serviceNamespace(ns)
        .resourceIds(tableId)
        .build();

    DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(response);
}

// Configure a scaling policy.
public static void configureScalingPolicy(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs, String policyName) {
    // Check if the policy exists before creating a new one.
```

```
DescribeScalingPoliciesResponse describeScalingPoliciesResponse =
appAutoScalingClient.describeScalingPolicies(DescribeScalingPoliciesRequest.builder()
    .serviceNamespace(ns)
    .resourceId(tableId)
    .scalableDimension(tableWCUs)
    .build());

if (!describeScalingPoliciesResponse.scalingPolicies().isEmpty()) {
    // If policies exist, consider updating an existing policy instead of
    creating a new one.
    System.out.println("Policy already exists. Consider updating it
instead.");
    List<ScalingPolicy> polList =
describeScalingPoliciesResponse.scalingPolicies();
    for (ScalingPolicy pol : polList) {
        System.out.println("Policy name:" +pol.policyName());
    }
} else {
    // If no policies exist, proceed with creating a new policy.
    PredefinedMetricSpecification specification =
PredefinedMetricSpecification.builder()

.predefinedMetricType(MetricType.DYNAMO_DB_WRITE_CAPACITY_UTILIZATION)
    .build();

    TargetTrackingScalingPolicyConfiguration policyConfiguration =
TargetTrackingScalingPolicyConfiguration.builder()
    .predefinedMetricSpecification(specification)
    .targetValue(50.0)
    .scaleInCooldown(60)
    .scaleOutCooldown(60)
    .build();

    PutScalingPolicyRequest putScalingPolicyRequest =
PutScalingPolicyRequest.builder()
    .targetTrackingScalingPolicyConfiguration(policyConfiguration)
    .serviceNamespace(ns)
    .scalableDimension(tableWCUs)
    .resourceId(tableId)
    .policyName(policyName)
    .policyType(PolicyType.TARGET_TRACKING_SCALING)
    .build();

    try {
```

```
        appAutoScalingClient.putScalingPolicy(putScalingPolicyRequest);
        System.out.println("You have successfully created a scaling policy
for an Application Auto Scaling scalable target");
    } catch (ApplicationAutoScalingException e) {
        System.err.println("Error: " + e.awsErrorDetails().errorMessage());
    }
}
}
```

- API の詳細については、「API リファレンス [RegisterScalableTarget](#)」の「」を参照してください。AWS SDK for Java 2.x

## SDK for Java 2.x を使用した Application Recovery Controller の例

次のコード例は、Application Recovery Controller AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- [アクション](#)

アクション

### GetRoutingControlState

次の例は、GetRoutingControlState を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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;
}
```

- API の詳細については、「API リファレンス [GetRoutingControlState](#)」の「」を参照してください。 AWS SDK for Java 2.x

## UpdateRoutingControlState

次の例は、UpdateRoutingControlState を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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;
}
```

- API の詳細については、「API リファレンス [UpdateRoutingControlState](#)」の「」を参照してください。AWS SDK for Java 2.x

## SDK for Java 2.x を使用した Aurora の例

次のコード例は、Aurora AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

### 開始方法

Hello Aurora

次のコード例は、Aurora の使用を開始する方法を示しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
```

```
RdsClient rdsClient = RdsClient.builder()
    .region(region)
    .build();

describeClusters(rdsClient);
rdsClient.close();
}

public static void describeClusters(RdsClient rdsClient) {
    DescribeDBClustersIterable clustersIterable =
rdsClient.describeDBClustersPaginator();
    clustersIterable.stream()
        .flatMap(r -> r.dbClusters().stream())
        .forEach(cluster -> System.out
            .println("Database name: " + cluster.databaseName() + " Arn
= " + cluster.dbClusterArn()));
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DescribeDBClusters](#)」を参照してください。

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### CreateDBCluster

次の例は、CreateDBCluster を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
    String dbClusterIdentifier, String userName, String password) {
    try {
        CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
            .databaseName(dbName)
            .dbClusterIdentifier(dbClusterIdentifier)
            .dbClusterParameterGroupName(dbParameterGroupFamily)
            .engine("aurora-mysql")
            .masterUsername(userName)
            .masterUserPassword(password)
            .build();

        CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
        return response.dbCluster().dbClusterArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[CreateDBCluster](#)」を参照してください。

## CreateDBClusterParameterGroup

次の例は、CreateDBClusterParameterGroup を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。



```
public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
    String dbParameterGroupFamily) {
    try {
        CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [CreateDBClusterParameterGroup](#)」を参照してください。AWS SDK for Java 2.x

## CreateDBClusterSnapshot

次の例は、CreateDBClusterSnapshot を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
    String dbSnapshotIdentifier) {
```

```
try {
    CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
        .build();

    CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
    System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス [CreateDBClusterSnapshot](#)」を参照してください。AWS SDK for Java 2.x

## CreateDBInstance

次の例は、CreateDBInstance を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String createDBInstanceCluster(RdsClient rdsClient,
    String dbInstanceIdentifier,
    String dbInstanceClusterIdentifier,
    String instanceClass) {
    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
```

```
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .engine("aurora-mysql")
        .dbInstanceClass(instanceClass)
        .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[CreateDBInstance](#)」を参照してください。

## DeleteDBCluster

次の例は、DeleteDBCluster を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
    try {
        DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
```

```
        .skipFinalSnapshot(true)
        .build();

        rdsClient.deleteDBCluster(deleteDbClusterRequest);
        System.out.println(dbInstanceClusterIdentifier + " was deleted!");
    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DeleteDBCluster](#)」を参照してください。

## DeleteDBClusterParameterGroup

次の例は、DeleteDBClusterParameterGroup を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
    throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
```

```

        int listSize = instanceList.size();
        didFind = false;
        int index = 1;
        for (DBInstance instance : instanceList) {
            instanceARN = instance.dbInstanceArn();
            if (instanceARN.compareTo(clusterDBARN) == 0) {
                System.out.println(clusterDBARN + " still exists");
                didFind = true;
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.
                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }

    DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
        .builder()
        .dbClusterParameterGroupName(dbClusterGroupName)
        .build();

    rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
    System.out.println(dbClusterGroupName + " was deleted.");

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

```

- API の詳細については、「API リファレンス [DeleteDBClusterParameterGroup](#)」を参照してください。AWS SDK for Java 2.x

## DeleteDBInstance

次の例は、DeleteDBInstance を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .deleteAutomatedBackups(true)
            .skipFinalSnapshot(true)
            .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DeleteDBInstance](#)」を参照してください。

## DescribeDBClusterParameterGroups

次の例は、DescribeDBClusterParameterGroups を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
        DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .maxRecords(20)
            .build();

        List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
            .dbClusterParameterGroups();
        for (DBClusterParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbClusterParameterGroupName());
            System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API [DescribeDBClusterParameterGroups](#) AWS SDK for Java 2.x」を参照してください。

## DescribeDBClusterParameters

次の例は、DescribeDBClusterParameters を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void describeDbClusterParameters(RdsClient rdsClient, String
dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .build();
        } else {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .source("user")
                .build();
        }

        DescribeDbClusterParametersResponse response = rdsClient
            .describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
            if ((paraName.compareTo("auto_increment_offset") == 0)
                || (paraName.compareTo("auto_increment_increment ") == 0)) {
                System.out.println("*** The parameter name is " + paraName);
                System.out.println("*** The parameter value is " +
para.parameterValue());
                System.out.println("*** The parameter data type is " +
para.dataType());
                System.out.println("*** The parameter description is " +
para.description());
            }
        }
    }
}
```



```
        System.out.println("*** The parameter allowed values is " +
para.allowedValues());
    }
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API [DescribeDBClusterParameters](#) AWS SDK for Java 2.x」を参照してください。

## DescribeDBClusterSnapshots

次の例は、DescribeDBClusterSnapshots を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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(snapshotRequest);
            List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
            for (DBClusterSnapshot snapshot : snapshotList) {
                snapshotReadyStr = snapshot.status();
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true;
                } else {
                    System.out.println(".");
                    Thread.sleep(sleepTime * 5000);
                }
            }
        }

        System.out.println("The Snapshot is available!");

    } catch (RdsException | InterruptedException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API [DescribeDBClusterSnapshots](#) AWS SDK for Java 2.x 」を参照してください。

## DescribeDBClusters

次の例は、DescribeDBClusters を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void describeDbClusterParameters(RdsClient rdsClient, String
dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .build();
        } else {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .source("user")
                .build();
        }

        DescribeDbClusterParametersResponse response = rdsClient
            .describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
            if ((paraName.compareTo("auto_increment_offset") == 0)
                || (paraName.compareTo("auto_increment_increment ") == 0)) {
                System.out.println("*** The parameter name is " + paraName);
                System.out.println("*** The parameter value is " +
para.parameterValue());
                System.out.println("*** The parameter data type is " +
para.dataType());
                System.out.println("*** The parameter description is " +
para.description());
                System.out.println("*** The parameter allowed values is " +
para.allowedValues());
            }
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DescribeDBClusters](#)」を参照してください。

## DescribeDBEngineVersions

次の例は、DescribeDBEngineVersions を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .engine("aurora-mysql")
            .defaultOnly(true)
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engineOb : engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "
                + engineOb.dbParameterGroupFamily());
            System.out.println("The name of the database engine " +
engineOb.engine());
            System.out.println("The version number of the database engine " +
engineOb.engineVersion());
        }
    }
}
```

```
    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API [DescribeDBEngineVersions](#) AWS SDK for Java 2.x」を参照してください。

## DescribeDBInstances

次の例は、DescribeDBInstances を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
            .dbClusterIdentifier(dbClusterIdentifier)
            .build();

        while (!instanceReady) {
            DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
            List<DBCluster> clusterList = response.dbClusters();
            for (DBCluster cluster : clusterList) {
                instanceReadyStr = cluster.status();
            }
        }
    }
}
```

```
        if (instanceReadyStr.contains("available")) {
            instanceReady = true;
        } else {
            System.out.print(".");
            Thread.sleep(sleepTime * 1000);
        }
    }
}
System.out.println("Database cluster is available!");

} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DescribeDBInstances](#)」を参照してください。

## DescribeOrderableDBInstanceOptions

次の例は、DescribeOrderableDBInstanceOptions を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .engine("aurora-mysql")
            .defaultOnly(true)
            .maxRecords(20)
            .build();
```

```
DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
List<DBEngineVersion> engines = response.dbEngineVersions();

// Get all DBEngineVersion objects.
for (DBEngineVersion engineOb : engines) {
    System.out.println("The name of the DB parameter group family for
the database engine is "
        + engineOb.dbParameterGroupFamily());
    System.out.println("The name of the database engine " +
engineOb.engine());
    System.out.println("The version number of the database engine " +
engineOb.engineVersion());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- APIの詳細については、「API リファレンス」の[DescribeOrderable「DBInstanceOptions」](#)を参照してください。AWS SDK for Java 2.x

## ModifyDBClusterParameterGroup

次の例は、ModifyDBClusterParameterGroup を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
```

```
DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
    .dbClusterParameterGroupName(dbClusterGroupName)
    .maxRecords(20)
    .build();

List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
    .dbClusterParameterGroups();
for (DBClusterParameterGroup group : groups) {
    System.out.println("The group name is " +
group.dbClusterParameterGroupName());
    System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス [ModifyDBClusterParameterGroup](#)」を参照してください。AWS SDK for Java 2.x

## シナリオ

### DB クラスターの開始方法

次のコードサンプルは、以下の操作方法を示しています。

- カスタム Aurora DB クラスターパラメータグループを作成し、パラメータ値を設定します。
- パラメータグループを使用する DB クラスターを作成する
- データベースを含む DB インスタンスを作成します。
- DB クラスターのスナップショットを作成して、リソースをクリーンアップします。



## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating\_how-services-use-secrets\_RS.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition
 * by calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.
 * 2. Selects an engine family and creates a custom DB cluster parameter group
 * by invoking the describeDBClusterParameters method.
 * 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups
 * method.
 * 4. Gets parameters in the group by invoking the describeDBClusterParameters
 * method.
 * 5. Modifies the auto_increment_offset parameter by invoking the
 * modifyDbClusterParameterGroupRequest method.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions by invoking the
 * describeDbEngineVersions method.
 * 8. Creates an Aurora DB cluster database cluster that contains a MySQL
 * database.
 * 9. Waits for DB instance to be ready.
 * 10. Gets a list of instance classes available for the selected engine.
```

```

* 11. Creates a database instance in the cluster.
* 12. Waits for DB instance to be ready.
* 13. Creates a snapshot.
* 14. Waits for DB snapshot to be ready.
* 15. Deletes the DB cluster.
* 16. Deletes the DB cluster group.
*/
public class AuroraScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
            "Usage:\n" +
            "    <dbClusterGroupName> <dbParameterGroupFamily>
<dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName>
<dbSnapshotIdentifier><secretName>"
            +
            "Where:\n" +
            "    dbClusterGroupName - The name of the DB cluster parameter
group. \n" +
            "    dbParameterGroupFamily - The DB cluster parameter group family
name (for example, aurora-mysql5.7). \n"
            +
            "    dbInstanceClusterIdentifier - The instance cluster identifier
value.\n" +
            "    dbInstanceIdentifier - The database instance identifier.\n" +
            "    dbName - The database name.\n" +
            "    dbSnapshotIdentifier - The snapshot identifier.\n" +
            "    secretName - The name of the AWS Secrets Manager secret that
contains the database credentials\`\n";
        ;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbClusterGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceClusterIdentifier = args[2];
        String dbInstanceIdentifier = args[3];
        String dbName = args[4];
        String dbSnapshotIdentifier = args[5];

```

```
String secretName = args[6];

// Retrieve the database credentials using AWS Secrets Manager.
Gson gson = new Gson();
User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
String username = user.getUsername();
String userPassword = user.getPassword();

Region region = Region.US_WEST_2;
RdsClient rdsClient = RdsClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon Aurora example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Return a list of the available DB engines");
describeDBEngines(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a custom parameter group");
createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get the parameter group");
describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get the parameters in the group");
describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Modify the auto_increment_offset parameter");
modifyDBClusterParas(rdsClient, dbClusterGroupName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("6. Display the updated parameter value");
describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of allowed engine versions");
getAllowedEngines(rdsClient, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Create an Aurora DB cluster database");
String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName,
dbName, dbInstanceClusterIdentifier,
    username, userPassword);
System.out.println("The ARN of the cluster is " + arnClusterVal);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Wait for DB instance to be ready");
waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get a list of instance classes available for the
selected engine");
String instanceClass = getListInstanceClasses(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Create a database instance in the cluster.");
String clusterDBARN = createDBInstanceCluster(rdsClient,
dbInstanceIdentifier, dbInstanceClusterIdentifier,
    instanceClass);
System.out.println("The ARN of the database is " + clusterDBARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Wait for DB instance to be ready");
waitDBInstanceReady(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Create a snapshot");
```

```
        createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Delete the DB cluster");
        deleteCluster(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete the DB cluster group");
        deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
            .region(region)

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

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
```

```
        .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
        throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(clusterDBARN) == 0) {
                    System.out.println(clusterDBARN + " still exists");
                    didFind = true;
                }
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.

                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }

        DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
            .builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .build();
```

```
        rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
        System.out.println(dbClusterGroupName + " was deleted.");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
    try {
        DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .skipFinalSnapshot(true)
            .build();

        rdsClient.deleteDBCluster(deleteDbClusterRequest);
        System.out.println(dbInstanceClusterIdentifier + " was deleted!");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .deleteAutomatedBackups(true)
            .skipFinalSnapshot(true)
            .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
```

```
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
    String dbInstanceClusterIdentifier) {
    try {
        boolean snapshotReady = false;
        String snapshotReadyStr;
        System.out.println("Waiting for the snapshot to become available.");

        DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .build();

        while (!snapshotReady) {
            DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
            List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
            for (DBClusterSnapshot snapshot : snapshotList) {
                snapshotReadyStr = snapshot.status();
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true;
                } else {
                    System.out.println(".");
                    Thread.sleep(sleepTime * 5000);
                }
            }
        }

        System.out.println("The Snapshot is available!");

    } catch (RdsException | InterruptedException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```



```
public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
    String dbSnapshotIdentifier) {
    try {
        CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
        System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void waitDBInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        String endpoint = "";
        while (!instanceReady) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                instanceReadyStr = instance.dbInstanceStatus();
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint().address();
                    instanceReady = true;
                } else {
                    System.out.print(".");
                }
            }
        }
    }
}
```

```
        Thread.sleep(sleepTime * 1000);
    }
}
}
System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static String createDBInstanceCluster(RdsClient rdsClient,
    String dbInstanceIdentifier,
    String dbInstanceClusterIdentifier,
    String instanceClass) {
    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .engine("aurora-mysql")
        .dbInstanceClass(instanceClass)
        .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static String getListInstanceClasses(RdsClient rdsClient) {
    try {
        DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest
        .builder()
```

```

        .engine("aurora-mysql")
        .maxRecords(20)
        .build();

DescribeOrderableDbInstanceOptionsResponse response = rdsClient
    .describeOrderableDBInstanceOptions(optionsRequest);
List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
String instanceClass = "";
for (OrderableDBInstanceOption instanceOption : instanceOptions) {
    instanceClass = instanceOption.dbInstanceClass();
    System.out.println("The instance class is " +
instanceOption.dbInstanceClass());
    System.out.println("The engine version is " +
instanceOption.engineVersion());
}
return instanceClass;

} catch (RdsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
            .dbClusterIdentifier(dbClusterIdentifier)
            .build();

        while (!instanceReady) {
            DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
            List<DBCluster> clusterList = response.dbClusters();
            for (DBCluster cluster : clusterList) {
                instanceReadyStr = cluster.status();
                if (instanceReadyStr.contains("available")) {

```

```
        instanceReady = true;
    } else {
        System.out.print(".");
        Thread.sleep(sleepTime * 1000);
    }
}
}
System.out.println("Database cluster is available!");

} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
String dbClusterIdentifier, String userName, String password) {
    try {
        CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
            .databaseName(dbName)
            .dbClusterIdentifier(dbClusterIdentifier)
            .dbClusterParameterGroupName(dbParameterGroupFamily)
            .engine("aurora-mysql")
            .masterUsername(userName)
            .masterUserPassword(password)
            .build();

        CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
        return response.dbCluster().dbClusterArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
```

```
        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
    .dbParameterGroupFamily(dbParameterGroupFamily)
    .engine("aurora-mysql")
    .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
        List<DBEngineVersion> dbEngines = response.dbEngineVersions();
        for (DBEngineVersion dbEngine : dbEngines) {
            System.out.println("The engine version is " +
dbEngine.engineVersion());
            System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Modify the auto_increment_offset parameter.
public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();

        List<Parameter> paraList = new ArrayList<>();
        paraList.add(parameter1);
        ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
            .dbClusterParameterGroupName(dClusterGroupName)
            .parameters(paraList)
            .build();

        ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
        System.out.println(
```

```
        "The parameter group " + response.dbClusterParameterGroupName()
+ " was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDbClusterParameters(RdsClient rdsClient, String
dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .build();
        } else {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .source("user")
                .build();
        }

        DescribeDbClusterParametersResponse response = rdsClient
            .describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
            if ((paraName.compareTo("auto_increment_offset") == 0)
                || (paraName.compareTo("auto_increment_increment ") == 0)) {
                System.out.println("*** The parameter name is " + paraName);
                System.out.println("*** The parameter value is " +
para.parameterValue());
                System.out.println("*** The parameter data type is " +
para.dataType());
                System.out.println("*** The parameter description is " +
para.description());
            }
        }
    }
}
```

```
        System.out.println("*** The parameter allowed values is " +
para.allowedValues());
    }
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
        DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .maxRecords(20)
            .build();

        List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
            .dbClusterParameterGroups();
        for (DBClusterParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbClusterParameterGroupName());
            System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
String dbParameterGroupFamily) {
    try {
        CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
```

```
        .description("Created by using the AWS SDK for Java")
        .build();

        CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .engine("aurora-mysql")
            .defaultOnly(true)
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engineOb : engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "
                + engineOb.dbParameterGroupFamily());
            System.out.println("The name of the database engine " +
engineOb.engine());
            System.out.println("The version number of the database engine " +
engineOb.engineVersion());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
```



- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [CreateDBCluster](#)
  - [CreateDBClusterParameterGroup](#)
  - [CreateDBClusterSnapshot](#)
  - [CreateDBInstance](#)
  - [DeleteDBCluster](#)
  - [DeleteDBClusterParameterGroup](#)
  - [DeleteDBInstance](#)
  - [DescribeDBClusterParameterGroups](#)
  - [DescribeDBClusterParameters](#)
  - [DescribeDBClusterSnapshots](#)
  - [DescribeDBClusters](#)
  - [DescribeDBEngineVersions](#)
  - [DescribeDBInstances](#)
  - [DescribeOrderableDBInstanceOptions](#)
  - [ModifyDBClusterParameterGroup](#)

## SDK for Java 2.x を使用した Auto Scaling の例

次のコード例は、Auto Scaling AWS SDK for Java 2.x でを使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

## 開始方法

こんにちは、Auto Scaling

次のコード例は、Auto Scaling の使用を開始する方法を示しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        describeGroups(autoScalingClient);
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
```

```
        System.out.println("Group Name: " + group.autoScalingGroupName());
        System.out.println("Group ARN: " + group.autoScalingGroupARN());
    });
}
}
```

- API の詳細については、「API リファレンス [DescribeAutoScalingGroups](#)」の「」を参照してください。AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### CreateAutoScalingGroup

次の例は、CreateAutoScalingGroup を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [CreateAutoScalingGroup](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteAutoScalingGroup

次の例は、DeleteAutoScalingGroup を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「API リファレンス [DeleteAutoScalingGroup](#)」の「」を参照してください。AWS SDK for Java 2.x

## DescribeAutoScalingGroups

次の例は、DescribeAutoScalingGroups を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
    }
}
```

- API の詳細については、「API リファレンス[DescribeAutoScalingGroups](#)」の「」を参照してください。AWS SDK for Java 2.x

## DescribeAutoScalingInstances

次の例は、DescribeAutoScalingInstances を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [DescribeAutoScalingInstances](#)」の「」を参照してください。 AWS SDK for Java 2.x

## DescribeScalingActivities

次の例は、DescribeScalingActivities を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [DescribeScalingActivities](#)」の「」を参照してください。 AWS SDK for Java 2.x

## DisableMetricsCollection

次の例は、DisableMetricsCollection を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [DisableMetricsCollection](#)」の「」を参照してください。 AWS SDK for Java 2.x

## EnableMetricsCollection

次の例は、EnableMetricsCollection を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [EnableMetricsCollection](#)」の「」を参照してください。 AWS SDK for Java 2.x

## SetDesiredCapacity

次の例は、SetDesiredCapacity を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [SetDesiredCapacity](#)」の「」を参照してください。 AWS SDK for Java 2.x

**TerminateInstanceInAutoScalingGroup**

次の例は、`TerminateInstanceInAutoScalingGroup` を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [TerminateInstanceInAutoScalingGroup](#)」の「」を参照してください。AWS SDK for Java 2.x

## UpdateAutoScalingGroup

次の例は、UpdateAutoScalingGroup を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス[UpdateAutoScalingGroup](#)」の「」を参照してください。AWS SDK for Java 2.x

## シナリオ

### レジリエントなサービスの構築と管理

次のコード例は、本、映画、曲のレコメンデーションを返す負荷分散型ウェブサービスの作成方法を示しています。この例は、障害に対するサービスの対応方法と、障害発生時の耐障害性を高めるためにサービスを再構築する方法を示しています。



- Amazon EC2 Auto Scaling グループを使用して、起動テンプレートに基づいて Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを作成し、インスタンス数を所定の範囲内に維持します。
- Elastic Load Balancing で HTTP リクエストを処理して配信します。
- Auto Scaling グループ内のインスタンスの状態を監視し、正常なインスタンスにのみリクエストを転送します。
- 各 EC2 インスタンスで Python ウェブサーバーを実行して HTTP リクエストを処理します。ウェブサーバーはレコメンデーションとヘルスチェックを返します。
- Amazon DynamoDB テーブルを使用してレコメンデーションサービスをシミュレートできます。
- AWS Systems Manager パラメータを更新して、リクエストとヘルスチェックに対するウェブサーバーの応答を制御します。

## SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\\
\\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\\
\\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
```

```
public static final String policyName = "doc-example-resilience-pol";
public static final String profileName = "doc-example-resilience-prof";

public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

public static final String targetGroupName = "doc-example-resilience-tg";
public static final String autoScalingGroupName = "doc-example-resilience-
group";
public static final String lbName = "doc-example-resilience-lb";
public static final String protocol = "HTTP";
public static final int port = 80;

public static final String DASHES = new String(new char[80]).replace("\0", "-");

public static void main(String[] args) throws IOException, InterruptedException
{
    Scanner in = new Scanner(System.in);
    Database database = new Database();
    AutoScaler autoScaler = new AutoScaler();
    LoadBalancer loadBalancer = new LoadBalancer();

    System.out.println(DASHES);
    System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("A - SETUP THE RESOURCES");
    System.out.println("Press Enter when you're ready to start deploying
resources.");
    in.nextLine();
    deploy(loadBalancer);
    System.out.println(DASHES);
    System.out.println(DASHES);
    System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    demo(loadBalancer);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("C - DELETE THE RESOURCES");
    System.out.println(""
```

This concludes the demo of how to build and manage a resilient service.

To keep things tidy and to avoid unwanted charges on your account, we can clean up all AWS resources

that were created for this demo.

```
""");
```

```
System.out.println("\n Do you want to delete the resources (y/n)? ");
```

```
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input
```

```
if (userInput.equals("y")) {
```

```
    // Delete resources here
```

```
    deleteResources(loadBalancer, autoScaler, database);
```

```
    System.out.println("Resources deleted.");
```

```
} else {
```

```
    System.out.println("""
```

```
        Okay, we'll leave the resources intact.
```

```
        Don't forget to delete them when you're done with them or you
```

```
might incur unexpected charges.
```

```
        """);
```

```
}
```

```
System.out.println(DASHES);
```

```
System.out.println(DASHES);
```

```
System.out.println("The example has completed. ");
```

```
System.out.println("\n Thanks for watching!");
```

```
System.out.println(DASHES);
```

```
}
```

```
// Deletes the AWS resources used in this example.
```

```
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler autoScaler, Database database)
```

```
    throws IOException, InterruptedException {
```

```
    loadBalancer.deleteLoadBalancer(lbName);
```

```
    System.out.println("*** Wait 30 secs for resource to be deleted");
```

```
    TimeUnit.SECONDS.sleep(30);
```

```
    loadBalancer.deleteTargetGroup(targetGroupName);
```

```
    autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
```

```
    autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
```

```
    autoScaler.deleteTemplate(templateName);
```

```
    database.deleteTable(tableName);
```

```
}
```

```
private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
    System.out.println(
        """
        For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
        to set up a load-balanced web service endpoint and explore
some ways to make it resilient
        against various kinds of failures.

        Some of the resources create by this demo are:
        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
        \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
        """);

    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating and populating a DynamoDB table named " +
tableName);
    Database database = new Database();
    database.createTable(tableName, fileName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("""
        Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
        This script starts a Python web server defined in the `server.py`
script. The web server
        listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
        For demo purposes, this server is run as the root user. In
production, the best practice is to
        run a web server, such as Apache, with least-privileged credentials.
```

```
        The template also defines an IAM policy that each instance uses to
assume a role that grants
        permissions to access the DynamoDB recommendation table and Systems
Manager parameters
        that control the flow of the demo.
        """);
```

```
        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);
```

```
        System.out.println("""
            At this point, you have EC2 instances created. Once each instance
starts, it listens for
            HTTP requests. You can see these instances in the console or
continue with the demo.
            Press Enter when you're ready to continue.
            """);
```

```
        in.nextLine();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("""
            Creating an Elastic Load Balancing target group and load balancer.
The target group
```

```
        defines how the load balancer connects to instances. The load
balancer provides a
        single endpoint where clients connect and dispatches requests to
instances in the group.
        """);

    String vpcId = autoScaler.getDefaultVPC();
    List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
    System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
    String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
    String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
    autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
    System.out.println("Verifying access to the load balancer endpoint...");
    boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
    if (!wasSuccessful) {
        System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to "http://checkip.amazonaws.com"
        HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
        try {
            // Execute the request and get the response
            HttpResponse response = httpClient.execute(httpGet);

            // Read the response content.
            String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

            // Print the public IP address.
            System.out.println("Public IP Address: " + ipAddress);
            GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
            if (!groupInfo.isPortOpen()) {
                System.out.println("""
                    For this example to work, the default security group for
your default VPC must
                    allow access from this computer. You can either add it
automatically from this
```

```

        example or add it yourself using the AWS Management
Console.
        """);

        System.out.println(
            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
        System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
        String ans = in.nextLine();
        if ("y".equalsIgnoreCase(ans)) {
            autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
            System.out.println("Security group rule added.");
        } else {
            System.out.println("No security group rule added.");
        }
    }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
    } else if (wasSuccessful) {
        System.out.println("Your load balancer is ready. You can access it by
browsing to:");
        System.out.println("\t http://" + elbDnsName);
    } else {
        System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
        System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
        System.out.println("you can successfully make a GET request to the load
balancer.");
    }

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();

```

```
System.out.println("Read the ssm_only_policy.json file");
String ssmOnlyPolicy = readFileAsString(ssmJSON);

System.out.println("Resetting parameters to starting values for demo.");
paramHelper.reset();

System.out.println(
    """
        This part of the demonstration shows how to toggle
different parts of the system
        to create situations where the web service fails, and shows
how using a resilient
        architecture can keep the web service running in spite of
these failures.

        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
    """);
demoChoices(loadBalancer);

System.out.println(
    """
        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
    """);
paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

System.out.println(
    """
        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
    """);
demoChoices(loadBalancer);

System.out.println(
    """
        Instead of failing when the recommendation service fails,
the web service can return a static response.
    """);
```



```
        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
        """);
    paramHelper.put(paramHelper.failureResponse, "static");

    System.out.println("""
        Now, sending a GET request to the load balancer endpoint returns a
static response.
        The service still reports as healthy because health checks are still
shallow.
        """);
    demoChoices(loadBalancer);

    System.out.println("Let's reinstate the recommendation service.");
    paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

    System.out.println("""
        Let's also substitute bad credentials for one of the instances in
the target group so that it can't
        access the DynamoDB recommendation table. We will get an instance id
value.
        """);

    LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
    AutoScaler autoScaler = new AutoScaler();

    // Create a new instance profile based on badCredsProfileName.
    templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
    String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
    System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

    String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
    System.out.println("The association Id value is " + profileAssociationId);
    System.out.println("Replacing the profile for instance " + badInstanceId
        + " with a profile that contains bad credentials");
    autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

    System.out.println(
        """"
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
```

```
        depending on which instance is selected by the load
balancer.
        """);

    demoChoices(loadBalancer);

    System.out.println("""
        Let's implement a deep health check. For this demo, a deep health
check tests whether
        the web service can access the DynamoDB table that it depends on for
recommendations. Note that
        the deep health check is only for ELB routing and not for Auto
Scaling instance health.
        This kind of deep health check is not recommended for Auto Scaling
instance health, because it
        risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
        """);

    System.out.println("""
        By implementing deep health checks, the load balancer can detect
when one of the instances is failing
        and take that instance out of rotation.
        """);

    paramHelper.put(paramHelper.healthCheck, "deep");

    System.out.println("""
        Now, checking target health indicates that the instance with bad
credentials
        is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
        instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
        the load balancer takes unhealthy instances out of its rotation.
        """);

    demoChoices(loadBalancer);

    System.out.println(
        ""
        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
```

```
        instance is to terminate it and let the auto scaler start a
new instance to replace it.
        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
            Even while the instance is terminating and the new instance is
starting, sending a GET
            request to the web service continues to get a successful
recommendation response because
            the load balancer routes requests to the healthy instances. After
the replacement instance
            starts and reports as healthy, it is included in the load balancing
rotation.

            Note that terminating and replacing an instance typically takes
several minutes, during which time you
            can see the changing health check status until the new instance is
running and healthy.
        """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
```

```
        System.out.println(i + ": " + actions[i]);
    }

    try {
        System.out.print("\nWhich action would you like to take? ");
        int choice = scanner.nextInt();
        System.out.println("-".repeat(88));

        switch (choice) {
            case 0 -> {
                System.out.println("Request:\n");
                System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                CloseableHttpClient httpClient =
HttpClientClients.createDefault();

                // Create an HTTP GET request to the ELB.
                HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);

                // Display the JSON response
                BufferedReader reader = new BufferedReader(
                    new
InputStreamReader(response.getEntity().getContent()));
                StringBuilder jsonResponse = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null) {
                    jsonResponse.append(line);
                }
                reader.close();

                // Print the formatted JSON response.
                System.out.println("Full Response:\n");
                System.out.println(jsonResponse.toString());

                // Close the HTTP client.
                httpClient.close();
            }
        }
    }
```

```

        case 1 -> {
            System.out.println("\nChecking the health of load balancer
targets:\n");
            List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
            for (TargetHealthDescription target : health) {
                System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
            }
            System.out.println("""
Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
        }
        case 2 -> {
            System.out.println("\nOkay, let's move on.");
            System.out.println("-".repeat(88));
            return; // Exit the method when choice is 2
        }
        default -> System.out.println("You must choose a value between
0-2. Please select again.");
    }

    } catch (java.util.InputMismatchException e) {
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}

```

Auto Scaling と Amazon EC2 のアクションをラップするクラスを作成します。

```
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.getInstanceProfile().getInstanceProfileName();
            System.out.println(name);
        }
    }

```

```
        RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
    .instanceProfileName(profileName)
    .roleName(roleName)
    .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(profileName)
    .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
    .roleName(roleName)
    .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
    .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
    .roleName(roleName)
    .policyArn(attachedPolicy.policyArn())
    .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");

    } catch (IamException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}

/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network, you
 * must instead specify a prefix list ID. You can also temporarily open the port
 * to
 * any IP address while running this example. If you do, be sure to remove
 * public
 * access when you're done.
 *
 */
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();
```

```

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
    .filters(filter, filter1)
    .build();

        DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
    .describeSecurityGroups(securityGroupsRequest);

        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " + secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " + ipPermission);
                    for (IpRange ipRange : ipPermission.ipRanges()) {
                        String cidrIp = ipRange.cidrIp();
                        if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                            System.out.println(cidrIp + " is applicable");
                            portIsOpen = true;
                        }
                    }

                    if (!ipPermission.prefixListIds().isEmpty()) {
                        System.out.println("Prefix lList is applicable");
                        portIsOpen = true;
                    }

                    if (!portIsOpen) {
                        System.out
                            .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                    } else {
                        break;
                    }
                }
            }
        }
    }
}

```

```
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }

    groupInfo.setPortOpen(portIsOpen);
    return groupInfo;
}

/**
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

    // Get availability zones.
    software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
            .builder()
            .build();
```

```
DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

String availabilityZones = String.join(",", availabilityZoneNames);
LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

String[] zones = availabilityZones.split(",");
CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

try {
    getAutoScalingClient().createAutoScalingGroup(groupRequest);
} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
return zones;
}

public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();
```

```
        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
            .builder()
            .filters(defaultFilter)
            .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
            .name("vpc-id")
            .values(vpcId)
            .build();

        Filter azFilter = Filter.builder()
            .name("availability-zone")
            .values(availabilityZones)
            .build();

        Filter defaultForAZ = Filter.builder()
            .name("default-for-az")
            .values("true")
            .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
            .filters(vpcFilter, azFilter, defaultForAZ)
            .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();
```

```
DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
List<String> instanceIds = autoScalingGroup.instances().stream()
    .map(instance -> instance.instanceId())
    .collect(Collectors.toList());

String[] instanceIdArray = instanceIds.toArray(new String[0]);
for (String instanceId : instanceIdArray) {
    System.out.println("Instance ID: " + instanceId);
    return instanceId;
}
return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
    ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
    ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
    for (Policy policy : listPoliciesResponse.policies()) {
        if (policy.policyName().equals(policyName)) {
            // List the entities (users, groups, roles) that are attached to the
policy.

```



```
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
    .builder()
    .policyArn(policy.arn())
    .build();
    ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
        .listEntitiesForPolicy(listEntitiesRequest);
    if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
        || !listEntitiesResponse.policyRoles().isEmpty()) {
        // Detach the policy from any entities it is attached to.
        DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
            .policyArn(policy.arn())
            .roleName(roleName) // Specify the name of the IAM role
            .build();

        getIAMClient().detachRolePolicy(detachPolicyRequest);
        System.out.println("Policy detached from entities.");
    }

    // Now, you can delete the policy.
    DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
        .policyArn(policy.arn())
        .build();

    getIAMClient().deletePolicy(deletePolicyRequest);
    System.out.println("Policy deleted successfully.");
    break;
}
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
```

```
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .roleName(roleName) // Remove the extra dot here
                .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
            .instanceProfileName(InstanceProfile)
            .build();

        getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Elastic Load Balancing のアクションをラップするクラスを作成します。

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
```

```
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
        .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
```

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

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {
            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);
            if (statusCode == 200) {
                success = true;
            } else {
                retries--;
            }
        }
    }
}
```

```
        System.out.println("Got connection error from load balancer
endpoint, retrying...");
        TimeUnit.SECONDS.sleep(15);
    }
}

} catch (org.apache.http.conn.HttpHostConnectException e) {
    System.out.println(e.getMessage());
}

System.out.println("Status.." + success);
return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}

/*
```

```
* Creates an Elastic Load Balancing load balancer that uses the specified
* subnets
* and forwards requests to the specified target group.
*/
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
```

```
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
```

DynamoDB を使用してレコメンデーションサービスをシミュレートするクラスを作成します。

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }
}
```

```
}

// Checks to see if the Amazon DynamoDB table exists.
private boolean doesTableExist(String tableName) {
    try {
        // Describe the table and catch any exceptions.
        DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        getDynamoDbClient().describeTable(describeTableRequest);
        System.out.println("Table '" + tableName + "' exists.");
        return true;
    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " + e.getMessage());
    }
    return false;
}

/**
 * Creates a DynamoDB table to use a recommendation service. The table has a
 * hash key named 'MediaType' that defines the type of media recommended, such
 * as
 * Book or Movie, and a range key named 'ItemId' that, combined with the
 * MediaType,
 * forms a unique identifier for the recommended item.
 */
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()

```



```

        .attributeName("ItemId")
        .attributeType(ScalarAttributeType.N)
        .build())
    .keySchema(
        KeySchemaElement.builder()
            .attributeName("MediaType")
            .keyType(KeyType.HASH)
            .build(),
        KeySchemaElement.builder()
            .attributeName("ItemId")
            .keyType(KeyType.RANGE)
            .build())
    .provisionedThroughput(
        ProvisionedThroughput.builder()
            .readCapacityUnits(5L)
            .writeCapacityUnits(5L)
            .build())
    .build();

    getDynamoDbClient().createTable(createTableRequest);
    System.out.println("Creating table " + tableName + "...");

    // Wait until the Amazon DynamoDB table is created.
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    WaiterResponse<DescribeTableResponse> waiterResponse =
    dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Table " + tableName + " created.");

    // Add records to the table.
    populateTable(fileName, tableName);
    }
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.

```

```
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
    System.out.println("Added all records to the " + tableName);
}
}
```

Systems Manager のアクションをラップするクラスを作成します。

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
    }
}
```

```
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [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](#)

## グループとインスタンスを管理する

次のコードサンプルは、以下の操作方法を示しています。

- 起動テンプレートとアベイラビリティゾーンを使用して、Amazon EC2 Auto Scaling グループを作成し、実行中のインスタンスに関する情報を取得します。
- Amazon CloudWatch メトリクスの収集を有効にします。
- グループの希望するキャパシティを更新し、インスタンスが起動するのを待ちます。
- グループ内の最も古いインスタンスを削除します。
- ユーザーのリクエストやキャパシティの変更に応じて発生するスケーリングアクティビティを一覧表示します。
- CloudWatch メトリクスの統計を取得してから、リソースをクリーンアップします。

## SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a launch template. For more information, see the
 * following topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-templates.html#create-launch-template
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
 * 13. Deletes the Auto Scaling group.
 */

public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <groupName> <launchTemplateName> <vpcZoneId>

            Where:
                groupName - The name of the Auto Scaling group.
                launchTemplateName - The name of the launch template.\s
    }
}
```

```
        vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
instances in the Auto Scaling group can be created.
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String groupName = args[0];
    String launchTemplateName = args[1];
    String vpcZoneId = args[2];
    AutoScalingClient autoScalingClient = AutoScalingClient.builder()
        .region(Region.US_EAST_1)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon EC2 Auto Scaling example
scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Create an Auto Scaling group named " + groupName);
    createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
    System.out.println(
        "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
    Thread.sleep(60000);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Get Auto Scale group Id value");
    String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupName);
    if (instanceId.compareTo("") == 0) {
        System.out.println("Error - no instance Id value");
        System.exit(1);
    } else {
        System.out.println("The instance Id value is " + instanceId);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
```

```
        System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
        describeAutoScalingInstance(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enable metrics collection " + instanceId);
        enableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update an Auto Scaling group to update max size to
3");
        updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Describe Auto Scaling groups");
        describeAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Describe account details");
        describeAccountLimits(autoScalingClient);
        System.out.println(
            "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Set desired capacity to 2");
        setDesiredCapacity(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get the two instance Id values and state");
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred for
the group");
        describeScalingActivities(autoScalingClient, groupName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group");
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                .autoScalingGroupName(groupName)
                .maxRecords(10)
                .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " + activity.activityId());
                System.out.println("The activity details are " +
activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```



```
        System.exit(1);
    }
}

public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
            .autoScalingGroupName(groupName)
            .desiredCapacity(2)
            .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
String launchTemplateName,
String vpcZoneId) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .availabilityZones("us-east-1a")
            .launchTemplate(templateSpecification)
            .maxSize(1)
            .minSize(1)
            .vpcZoneIdentifier(vpcZoneId)
            .build();

        autoScalingClient.createAutoScalingGroup(request);
    }
}
```

```
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
        .waitUntilGroupExists(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Auto Scaling Group created");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
        .builder()
        .instanceIds(id)
        .build();

        DescribeAutoScalingInstancesResponse response = autoScalingClient
        .describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance : instances) {
            System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
```

```
        try {
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .maxRecords(10)
                .build();

            DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("*** The service to use for the health checks: "
+ group.healthCheckType());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());
                List<Instance> instances = group.instances();

                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                    System.out.println("The instance id is " + instanceId);
                }
            }
        }
    }
}
```

```
        System.out.println("The lifecycle state is " +
instance.lifecycleState());
    }
}

    return instanceId;
} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .build();

        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
```

```
        System.out.println("The disable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAccountLimits(AutoScalingClient autoScalingClient) {
    try {
        DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
        System.out.println("The max number of auto scaling groups is " +
response.maxNumberOfAutoScalingGroups());
        System.out.println("The current number of auto scaling groups is " +
response.numberOfWorkingAutoScalingGroups());

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
    String launchTemplateName) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
            .maxSize(3)
            .autoScalingGroupName(groupName)
            .launchTemplate(templateSpecification)
            .build();

        autoScalingClient.updateAutoScalingGroup(groupRequest);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
```

```
        .autoScalingGroupNames(groupName)
        .build();

    WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
        .waitUntilGroupInService(groupsRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("You successfully updated the auto scaling group " +
groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance " + instanceId);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build();

        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    }
}
```

```
        System.out.println("You successfully deleted " + groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [CreateAutoScalingGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DescribeAutoScalingGroups](#)
  - [DescribeAutoScalingInstances](#)
  - [DescribeScalingActivities](#)
  - [DisableMetricsCollection](#)
  - [EnableMetricsCollection](#)
  - [SetDesiredCapacity](#)
  - [TerminateInstanceInAutoScalingGroup](#)
  - [UpdateAutoScalingGroup](#)

## SDK for Java 2.x を使用する Amazon Bedrock の例

次のコード例は、Amazon Bedrock AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

## トピック

- [アクション](#)

## アクション

### GetFoundationModel

次の例は、GetFoundationModel を使用する方法を説明しています。

#### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

同期 Amazon Bedrock クライアントを使用して基盤モデルの詳細を取得します。

```
/**
 * 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);
    }
}

```

非同期 Amazon Bedrock クライアントを使用して基盤モデルの詳細を取得します。

```

/**
 * Get details about an Amazon Bedrock foundation model.
 *
 * @param bedrockClient The async service client for accessing Amazon Bedrock.
 * @param modelIdentifier The model identifier.
 * @return An object containing the foundation model's details.
 */
public static FoundationModelDetails getFoundationModel(BedrockAsyncClient
bedrockClient, String modelIdentifier) {
    try {
        CompletableFuture<GetFoundationModelResponse> future =
bedrockClient.getFoundationModel(
            r -> r.modelIdentifier(modelIdentifier)
        );

        FoundationModelDetails model = future.get().modelDetails();
    }
}

```

```
        System.out.println(" Model ID: " + model.modelId());
        System.out.println(" Model ARN: " +
model.modelArn());
        System.out.println(" Model Name: " +
model.modelName());
        System.out.println(" Provider Name: " +
model.providerName());
        System.out.println(" Lifecycle status: " +
model.modelLifecycle().statusAsString());
        System.out.println(" Input modalities: " +
model.inputModalities());
        System.out.println(" Output modalities: " +
model.outputModalities());
        System.out.println(" Supported customizations: " +
model.customizationsSupported());
        System.out.println(" Supported inference types: " +
model.inferenceTypesSupported());
        System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

        return model;

    } catch (ExecutionException e) {
        if (e.getMessage().contains("ValidationException")) {
            throw new IllegalArgumentException(e.getMessage());
        } else {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}
```

- API の詳細については、「API リファレンス [GetFoundationModel](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListFoundationModels

次の例は、ListFoundationModels を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

同期 Amazon Bedrock クライアントを使用して、使用可能な Amazon Bedrock 基盤モデルを一覧表示します。

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

非同期 Amazon Bedrock クライアントを使用して、使用可能な Amazon Bedrock 基盤モデルを一覧表示します。

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

- API の詳細については、「API リファレンス [ListFoundationModels](#)」の「」を参照してください。AWS SDK for Java 2.x

## SDK for Java 2.x を使用する Amazon Bedrock ランタイムの例

次のコード例は、Amazon Bedrock ランタイム AWS SDK for Java 2.x でを使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### トピック

- [AI21 ラボ Jurassic-2](#)
- [Amazon Titan Image Generator](#)
- [Amazon Titan Text](#)
- [Amazon Titan Text Embeddings](#)
- [Anthropic Claude](#)
- [Cohere Command](#)
- [メタラマ](#)
- [ミスタル AI](#)
- [シナリオ](#)
- [Stable Diffusion](#)

## AI21 ラボ Jurassic-2

### 会話

次のコード例は、Bedrock の Converse API を使用して、AI21 Labs Jurassic-2 にテキストメッセージを送信する方法を示しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

Bedrock の Converse API を使用して、AI21 Labs Jurassic-2 にテキストメッセージを送信します。

```
// Use the Converse API to send a text message to AI21 Labs Jurassic-2.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Jurassic-2 Mid.
```

```
    var modelId = "ai21.j2-mid-v1";

    // Create the input text and embed it in a message object with the user
role.
    var inputText = "Describe the purpose of a 'hello world' program in one
line.";
    var message = Message.builder()
        .content(ContentBlock.fromText(inputText))
        .role(ConversationRole.USER)
        .build();

    try {
        // Send the message with a basic inference configuration.
        ConverseResponse response = client.converse(request -> request
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)));

        // Retrieve the generated text from Bedrock's response object.
        var responseText = response.output().message().content().get(0).text();
        System.out.println(responseText);

        return responseText;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converse();
}
}
```

非同期 Java クライアントで Bedrock の Converse API を使用して、AI21 Labs Jurassic-2 にテキストメッセージを送信します。

```
// Use the Converse API to send a text message to AI21 Labs Jurassic-2
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Jurassic-2 Mid.
        var modelId = "ai21.j2-mid-v1";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
```



```
        .temperature(0.5F)
        .topP(0.9F))
    );

    // Prepare a future object to handle the asynchronous response.
    CompletableFuture<String> future = new CompletableFuture<>();

    // Handle the response or error using the future object.
    request.whenComplete((response, error) -> {
        if (error == null) {
            // Extract the generated text from Bedrock's response object.
            String responseText =
response.output().message().content().get(0).text();
            future.complete(responseText);
        } else {
            future.completeExceptionally(error);
        }
    });

    try {
        // Wait for the future object to complete and retrieve the generated
text.
        String responseText = future.get();
        System.out.println(responseText);

        return responseText;
    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- API の詳細については、「[API リファレンス](#)」のAWS SDK for Java 2.x 「会話」を参照してください。

## InvokeModel

次のコード例は、Invoke Model API を使用して AI21 Labs Jurassic-2 にテキストメッセージを送信する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

モデル呼び出し API を使用してテキストメッセージを送信します。

```
// Use the native inference API to send a text message to AI21 Labs Jurassic-2.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Jurassic-2 Mid.
        var modelId = "ai21.j2-mid-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
```

```
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
jurassic2.html
var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/completions/0/data/
text").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    invokeModel();
}
}
```

- APIの詳細については、「APIリファレンス[InvokeModel](#)」の「」を参照してください。AWS SDK for Java 2.x

## Amazon Titan Image Generator

### InvokeModel

次のコード例は、Amazon Bedrock で Amazon Titan Image を呼び出してイメージを生成する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

Amazon Titan Image Generator を使用してイメージを作成します。

```
// Create an image with the Amazon Titan Image Generator.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```

```
// Set the model ID, e.g., Titan Image G1.
var modelId = "amazon.titan-image-generator-v1";

// The InvokeModel API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-image.html
var nativeRequestTemplate = """
    {
        "taskType": "TEXT_IMAGE",
        "textToImageParams": { "text": "{{prompt}}" },
        "imageGenerationConfig": { "seed": {{seed}} }
    }""";

// Define the prompt for the image generation.
var prompt = "A stylized picture of a cute old steampunk robot";

// Get a random 31-bit seed for the image generation (max. 2,147,483,647).
var seed = new BigInteger(31, new SecureRandom());

// Embed the prompt and seed in the model's native request payload.
var nativeRequest = nativeRequestTemplate
    .replace("{{prompt}}", prompt)
    .replace("{{seed}}", seed.toString());

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated image data from the model's response.
    var base64ImageData = new JSONPointer("/
images/0").queryFrom(responseBody).toString();

    return base64ImageData;

} catch (SdkClientException e) {
```

```
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    System.out.println("Generating image. This may take a few seconds...");

    String base64ImageData = invokeModel();

    displayImage(base64ImageData);
}
}
```

- API の詳細については、「API リファレンス [InvokeModel](#)」の「」を参照してください。AWS SDK for Java 2.x

## Amazon Titan Text

### 会話

次のコード例は、Bedrock の Converse API を使用して Amazon Titan Text にテキストメッセージを送信する方法を示しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

Bedrock の Converse API を使用して、Amazon Titan Text にテキストメッセージを送信します。

```
// Use the Converse API to send a text message to Amazon Titan Text.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
```

```
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                    .maxTokens(512)
                    .temperature(0.5F)
                    .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);
        }
    }
}
```

```
        return responseText;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converse();
}
}
```

非同期 Java クライアントで Bedrock の Converse API を使用して、Amazon Titan Text にテキストメッセージを送信します。

```
// Use the Converse API to send a text message to Amazon Titan Text
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```



```
// Set the model ID, e.g., Titan Text Premier.
var modelId = "amazon.titan-text-premier-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
    // Wait for the future object to complete and retrieve the generated
text.
    String responseText = future.get();
    System.out.println(responseText);
}
```

```
        return responseText;

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- API の詳細については、「[API リファレンス](#)」の AWS SDK for Java 2.x 「会話」を参照してください。

## ConverseStream

次のコード例は、Bedrock の Converse API を使用して Amazon Titan Text にテキストメッセージを送信し、レスポンスストリームをリアルタイムで処理する方法を示しています。

## SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

Bedrock の Converse API を使用して Amazon Titan Text にテキストメッセージを送信し、レスポンスストリームをリアルタイムで処理します。

```
// Use the Converse API to send a text message to Amazon Titan Text
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
```

```
import
software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
            .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                .onContentBlockDelta(chunk -> {
                    String responseText = chunk.delta().text();
                    System.out.print(responseText);
                }).build()
            ).onError(err ->
                System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
            ).build();

        try {
```

```
// Send the message with a basic inference configuration and attach the
handler.
client.converseStream(request -> request
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F)
    ), responseStreamHandler).get();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
            e.getCause().getMessage());
    }
}
}
```

- API の詳細については、「API リファレンス [ConverseStream](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## InvokeModel

次のコード例は、Invoke Model API を使用して Amazon Titan Text にテキストメッセージを送信する方法を示しています。

## SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

モデル呼び出し API を使用してテキストメッセージを送信します。

```
// Use the native inference API to send a text message to Amazon Titan Text.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
```

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        titan-text.html
        var nativeRequestTemplate = "{ \"inputText\": \"{{prompt}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
```

```
        var text = new JSONPointer("/results/0/outputText").queryFrom(responseBody).toString();
        System.out.println(text);

        return text;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    invokeModel();
}
}
```

- APIの詳細については、「API リファレンス [InvokeModel](#)」の「」を参照してください。AWS SDK for Java 2.x

## InvokeModelWithResponseStream

次のコード例は、Invoke Model API を使用して Amazon Titan Text モデルにテキストメッセージを送信し、レスポンスストリームを出力する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

モデル呼び出し API を使用してテキストメッセージを送信し、レスポンスストリームをリアルタイムで処理します。

```
// Use the native inference API to send a text message to Amazon Titan Text
// and print the response stream.
```

```
import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        titan-text.html
        var nativeRequestTemplate = "{ \"inputText\": \"{{prompt}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);
```

```
// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/outputText").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

try {
    // Send the request and wait for the handler to process the response.
    client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

    // Return the complete response text.
    return completeResponseTextBuffer.toString();

} catch (ExecutionException | InterruptedException e) {
    System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```



```
}
```

- API の詳細については、「API リファレンス [InvokeModelWithResponseStream](#)」の「」を参照してください。AWS SDK for Java 2.x

## Amazon Titan Text Embeddings

### InvokeModel

次のコードサンプルは、以下の操作方法を示しています。

- 最初の埋め込みの作成を開始します。
- デイメンションの数と正規化を設定する埋め込みを作成します (V2 のみ)。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

Titan Text Embeddings V2 で最初の埋め込みを作成します。

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

ディメンションの数と正規化を設定する Titan Text Embeddings V2 を呼び出します。

```
/**
 * Invoke Amazon Titan Text Embeddings V2 with additional inference parameters.
 *
 * @param inputText - The text to convert to an embedding.
 * @param dimensions - The number of dimensions the output embeddings should
have.
 *
 * Values accepted by the model: 256, 512, 1024.
 * @param normalize - A flag indicating whether or not to normalize the output
embeddings.
 * @return The {@link JSONObject} representing the model's response.
 */
public static JSONObject invokeModel(String inputText, int dimensions, boolean
normalize) {

    // Create a Bedrock Runtime client in the AWS Region of your choice.
    var client = BedrockRuntimeClient.builder()
        .region(Region.US_WEST_2)
        .build();

    // Set the model ID, e.g., Titan Embed Text v2.0.
    var modelId = "amazon.titan-embed-text-v2:0";

    // Create the request for the model.
    var nativeRequest = ""
        {
            "inputText": "%s",
            "dimensions": %d,
            "normalize": %b
        }
```

```
    }
    """).formatted(inputText, dimensions, normalize);

    // Encode and send the request.
    var response = client.invokeModel(request -> {
        request.body(SdkBytes.fromUtf8String(nativeRequest));
        request.modelId(modelId);
    });

    // Decode the model's response.
    var modelResponse = new JSONObject(response.body().asUtf8String());

    // Extract and print the generated embedding and the input text token count.
    var embedding = modelResponse.getJSONArray("embedding");
    var inputTokenCount = modelResponse.getBigInteger("inputTextTokenCount");
    System.out.println("Embedding: " + embedding);
    System.out.println("\nInput token count: " + inputTokenCount);

    // Return the model's native response.
    return modelResponse;
}
```

- API の詳細については、「API リファレンス [InvokeModel](#)」の「」を参照してください。AWS SDK for Java 2.x

## Anthropic Claude

### 会話

次のコード例は、Bedrock の Converse API を使用して Anthropic Claude にテキストメッセージを送信する方法を示しています。

SDK for Java 2.x

#### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

Bedrock の Converse API を使用して、Anthropic Claude にテキストメッセージを送信します。

```
// Use the Converse API to send a text message to Anthropic Claude.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                    .maxTokens(512)
```

```
        .temperature(0.5F)
        .topP(0.9F)));

    // Retrieve the generated text from Bedrock's response object.
    var responseText = response.output().message().content().get(0).text();
    System.out.println(responseText);

    return responseText;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    converse();
}
}
```

非同期 Java クライアントで Bedrock の Converse API を使用して、Anthropic Claude にテキストメッセージを送信します。

```
// Use the Converse API to send a text message to Anthropic Claude
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
// Replace the DefaultCredentialsProvider with your preferred credentials
provider.
var client = BedrockRuntimeAsyncClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Claude 3 Haiku.
var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});
```

```
        try {
            // Wait for the future object to complete and retrieve the generated
            text.

            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
    }
}
```

- API の詳細については、「API [リファレンス](#)」のAWS SDK for Java 2.x 「会話」を参照してください。

## ConverseStream

次のコード例は、Bedrock の Converse API を使用して Anthropic Claude にテキストメッセージを送信し、レスポンスストリームをリアルタイムで処理する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

Bedrock の Converse API を使用して Anthropic Claude にテキストメッセージを送信し、レスポンスストリームをリアルタイムで処理します。

```
// Use the Converse API to send a text message to Anthropic Claude
// and print the response stream.
```



```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
            .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                .onContentBlockDelta(chunk -> {
                    String responseText = chunk.delta().text();
                    System.out.print(responseText);
                }).build()
            ).onError(err ->
```

```
        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
            .build());

    try {
        // Send the message with a basic inference configuration and attach the
handler.
        client.converseStream(request -> request.modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)
            ), responseStreamHandler).get();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    }
}
}
```

- API の詳細については、「API リファレンス [ConverseStream](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## InvokeModel

次のコード例は、Invoke Model API を使用して Anthropic Claude にテキストメッセージを送信する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

モデル呼び出し API を使用してテキストメッセージを送信します。

```
// Use the native inference API to send a text message to Anthropic Claude.
```

```
import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        anthropic-claude-messages.html
        var nativeRequestTemplate = ""
            {
                "anthropic_version": "bedrock-2023-05-31",
                "max_tokens": 512,
                "temperature": 0.5,
                "messages": [{
                    "role": "user",
                    "content": "{{prompt}}"
                }]
            }"";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
```

```
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/content/0/
text").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    invokeModel();
}
}
```

- API の詳細については、「API リファレンス [InvokeModel](#)」の「」を参照してください。AWS SDK for Java 2.x

## InvokeModelWithResponseStream

次のコード例は、Invoke Model API を使用して Anthropic Claude モデルにテキストメッセージを送信し、レスポンスストリームを出力する方法を示しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

モデル呼び出し API を使用してテキストメッセージを送信し、レスポンスストリームをリアルタイムで処理します。

```
// Use the native inference API to send a text message to Anthropic Claude
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.Objects;
import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
```

```
        .build();

// Set the model ID, e.g., Claude 3 Haiku.
var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// The InvokeModelWithResponseStream API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
anthropic-claude-messages.html
var nativeRequestTemplate = """
    {
        "anthropic_version": "bedrock-2023-05-31",
        "max_tokens": 512,
        "temperature": 0.5,
        "messages": [{
            "role": "user",
            "content": "{{prompt}}"
        }]
    }""";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        var response = new JSONObject(chunk.bytes().asUtf8String());
```

```
        // Extract and print the text from the content blocks.
        if (Objects.equals(response.getString("type"),
"content_block_delta")) {
            var text = new JSONPointer("/delta/
text").queryFrom(response);
            System.out.print(text);

            // Append the text to the response text buffer.
            completeResponseTextBuffer.append(text);
        }
    }).build()).build();

    try {
        // Send the request and wait for the handler to process the response.
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- API の詳細については、「API リファレンス [InvokeModelWithResponseStream](#)」の「」を参照してください。AWS SDK for Java 2.x

## Cohere Command

会話: すべてのモデル

次のコード例は、Bedrock の Converse API を使用して Cohere Command にテキストメッセージを送信する方法を示しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

Bedrock の Converse API を使用して、Cohere コマンドにテキストメッセージを送信します。

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

非同期 Java クライアントで Bedrock の Converse API を使用して、Cohere Command にテキストメッセージを送信します。

```
// Use the Converse API to send a text message to Cohere Command
```

```
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F))
```

```
    );

    // Prepare a future object to handle the asynchronous response.
    CompletableFuture<String> future = new CompletableFuture<>();

    // Handle the response or error using the future object.
    request.whenComplete((response, error) -> {
        if (error == null) {
            // Extract the generated text from Bedrock's response object.
            String responseText =
response.output().message().content().get(0).text();
            future.complete(responseText);
        } else {
            future.completeExceptionally(error);
        }
    });

    try {
        // Wait for the future object to complete and retrieve the generated
text.
        String responseText = future.get();
        System.out.println(responseText);

        return responseText;
    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- API の詳細については、「[API リファレンス](#)」のAWS SDK for Java 2.x 「会話」を参照してください。

## ConverseStream: すべてのモデル

次のコード例は、Bedrock の Converse API を使用して Cohere Command にテキストメッセージを送信し、レスポンスストリームをリアルタイムで処理する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

Bedrock の Converse API を使用して Cohere Command にテキストメッセージを送信し、レスポンスストリームをリアルタイムで処理します。

```
// Use the Converse API to send a text message to Cohere Command
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```

```
// Set the model ID, e.g., Command R.
var modelId = "cohere.command-r-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Create a handler to extract and print the response text in real-time.
var responseStreamHandler = ConverseStreamResponseHandler.builder()
    .subscriber(ConverseStreamResponseHandler.Visitor.builder()
        .onContentBlockDelta(chunk -> {
            String responseText = chunk.delta().text();
            System.out.print(responseText);
        }).build())
    .onError(err ->
        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage()))
    ).build();

try {
    // Send the message with a basic inference configuration and attach the
handler.
    client.converseStream(request -> request.modelId(modelId)
        .messages(message)
        .inferenceConfig(config -> config
            .maxTokens(512)
            .temperature(0.5F)
            .topP(0.9F)
        ), responseStreamHandler).get();

} catch (ExecutionException | InterruptedException e) {
    System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
}
}
}
```

- API の詳細については、「API リファレンス [ConverseStream](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## InvokeModel: コマンド R と R+

次のコード例は、Invoke Model API を使用して Cohere Command R と R+ にテキストメッセージを送信する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

モデル呼び出し API を使用してテキストメッセージを送信します。

```
// Use the native inference API to send a text message to Cohere Command R.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Command_R_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";
```

```
// The InvokeModel API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command-r-plus.html
var nativeRequestTemplate = "{ \"message\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/text").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

public static void main(String[] args) {
    invokeModel();
}
}
```

- API の詳細については、「API リファレンス [InvokeModel](#)」の「」を参照してください。AWS SDK for Java 2.x

## InvokeModel: コマンドとコマンドライト

次のコード例は、Invoke Model API を使用して Cohere コマンドにテキストメッセージを送信する方法を示しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

モデル呼び出し API を使用してテキストメッセージを送信します。

```
// Use the native inference API to send a text message to Cohere Command.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Command_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command Light.
        var modelId = "cohere.command-light-text-v14";
```



```
// The InvokeModel API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command.html
var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/generations/0/
text").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

public static void main(String[] args) {
    invokeModel();
}
}
```

- API の詳細については、「API リファレンス [InvokeModel](#)」の「」を参照してください。AWS SDK for Java 2.x

### InvokeModelWithResponseStream: コマンド R と R+

次のコード例は、レスポンスストリームでモデル呼び出し API を使用して、Cohere コマンドにテキストメッセージを送信する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

モデル呼び出し API を使用してテキストメッセージを送信し、レスポンスストリームをリアルタイムで処理します。

```
// Use the native inference API to send a text message to Cohere Command R
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Command_R_InvokeModelWithResponseStream {
```

```
public static String invokeModelWithResponseStream() throws ExecutionException,
InterruptedException {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    // Replace the DefaultCredentialsProvider with your preferred credentials
    provider.
    var client = BedrockRuntimeAsyncClient.builder()
        .credentialsProvider(DefaultCredentialsProvider.create())
        .region(Region.US_EAST_1)
        .build();

    // Set the model ID, e.g., Command R.
    var modelId = "cohere.command-r-v1:0";

    // The InvokeModelWithResponseStream API uses the model's native payload.
    // Learn more about the available inference parameters and response fields
    at:
    // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
    cohere-command-r-plus.html
    var nativeRequestTemplate = "{ \"message\": \"{{prompt}}\" }";

    // Define the prompt for the model.
    var prompt = "Describe the purpose of a 'hello world' program in one line.";

    // Embed the prompt in the model's native request payload.
    String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

    // Create a request with the model ID and the model's native request
    payload.
    var request = InvokeModelWithResponseStreamRequest.builder()
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
        .build();

    // Prepare a buffer to accumulate the generated response text.
    var completeResponseTextBuffer = new StringBuilder();

    // Prepare a handler to extract, accumulate, and print the response text in
    real-time.
    var responseStreamHandler =
    InvokeModelWithResponseStreamResponseHandler.builder()
        .subscriber(Visitor.builder().onChunk(chunk -> {
            // Extract and print the text from the model's native response.
            var response = new JSONObject(chunk.bytes().asUtf8String());
```

```
        var text = new JSONPointer("/text").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

    try {
        // Send the request and wait for the handler to process the response.
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- API の詳細については、「API リファレンス [InvokeModel](#)」の「」を参照してください。AWS SDK for Java 2.x

### InvokeModelWithResponseStream: コマンドとコマンドライト

次のコード例は、レスポンスストリームでモデル呼び出し API を使用して、Cohere コマンドにテキストメッセージを送信する方法を示しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

モデル呼び出し API を使用してテキストメッセージを送信し、レスポンスストリームをリアルタイムで処理します。

```
// Use the native inference API to send a text message to Cohere Command
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Command_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```

```
// Set the model ID, e.g., Command Light.
var modelId = "cohere.command-light-text-v14";

// The InvokeModelWithResponseStream API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command.html
var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/generations/0/
text").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build());

try {
    // Send the request and wait for the handler to process the response.
```

```
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- API の詳細については、「API リファレンス [InvokeModel](#)」の「」を参照してください。AWS SDK for Java 2.x

## メタラマ

すべてのモデル: Converse API

次のコード例は、Bedrock の Converse API を使用して Meta Llama にテキストメッセージを送信する方法を示しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

Bedrock の Converse API を使用して、Meta Llama にテキストメッセージを送信します。

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

非同期 Java クライアントで Bedrock の Converse API を使用して、Meta Llama にテキストメッセージを送信します。

```
// Use the Converse API to send a text message to Meta Llama
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
```

```
        .credentialsProvider(DefaultCredentialsProvider.create())
        .region(Region.US_EAST_1)
        .build();

// Set the model ID, e.g., Llama 3 8b Instruct.
var modelId = "meta.llama3-8b-instruct-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
    // Wait for the future object to complete and retrieve the generated
text.
```

```
String responseText = future.get();
System.out.println(responseText);

return responseText;

} catch (ExecutionException | InterruptedException e) {
    System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
    throw new RuntimeException(e);
}
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- API の詳細については、「[API リファレンス](#)」の AWS SDK for Java 2.x 「会話」を参照してください。

## ConverseStream: すべてのモデル

次のコード例は、Bedrock の Converse API を使用してテキストメッセージを Meta Llama に送信し、レスポンスストリームをリアルタイムで処理する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

Bedrock の Converse API を使用して Meta Llama にテキストメッセージを送信し、レスポンスストリームをリアルタイムで処理します。

```
// Use the Converse API to send a text message to Meta Llama
// and print the response stream.

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

```
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
            .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                .onContentBlockDelta(chunk -> {
                    String responseText = chunk.delta().text();
                    System.out.print(responseText);
                }).build()
            ).onError(err ->
                System.err.printf("Can't invoke '%s': %s", modelId,
                err.getMessage())
            ).build();
```

```
    try {
        // Send the message with a basic inference configuration and attach the
handler.
        client.converseStream(request -> request
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)
            ), responseStreamHandler).get();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    }
}
}
```

- API の詳細については、「API リファレンス [ConverseStream](#)」の「」を参照してください。  
AWS SDK for Java 2.x

InvokeModel: ラマ 2

次のコード例は、モデル呼び出し API を使用して Meta Llama 2 にテキストメッセージを送信する方法を示しています。

SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

モデル呼び出し API を使用してテキストメッセージを送信します。

```
// Use the native inference API to send a text message to Meta Llama 2.

import org.json.JSONObject;
```

```
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Llama2_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 2 Chat 13B.
        var modelId = "meta.llama2-13b-chat-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        meta.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\n\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Llama 2's instruction format.
        var instruction = "<s>[INST] {{prompt}} [/INST]\n".replace("{{prompt}}",
        prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
        instruction);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                .body(SdkBytes.fromUtf8String(nativeRequest)))
```

```
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/
generation").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    invokeModel();
}
}
```

- APIの詳細については、「APIリファレンス[InvokeModel](#)」の「」を参照してください。AWS SDK for Java 2.x

### InvokeModel: ラマ 3

次のコード例は、モデル呼び出し API を使用して Meta Llama 3 にテキストメッセージを送信する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

モデル呼び出し API を使用してテキストメッセージを送信します。

```
// Use the native inference API to send a text message to Meta Llama 3.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Llama3_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        meta.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\n }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Llama 3's instruction format.
        var instruction = (
            "<|begin_of_text|>\n" +
            "<|start_header_id|>user<|end_header_id|>\n" +
            "{{prompt}} <|eot_id|>\n" +
            "<|start_header_id|>assistant<|end_header_id|>\n"
        ).replace("{{prompt}}", prompt);
```



```
// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/
generation").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

public static void main(String[] args) {
    invokeModel();
}
}
```

- API の詳細については、「API リファレンス [InvokeModel](#)」の「」を参照してください。AWS SDK for Java 2.x

## InvokeModelWithResponseStream: ラマ 2

次のコード例は、モデル呼び出し API を使用して Meta Llama 2 にテキストメッセージを送信し、レスポンスストリームを出力する方法を示しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

モデル呼び出し API を使用してテキストメッセージを送信し、レスポンスストリームをリアルタイムで処理します。

```
// Use the native inference API to send a text message to Meta Llama 2
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Llama2_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```

```
// Set the model ID, e.g., Llama 2 Chat 13B.
var modelId = "meta.llama2-13b-chat-v1";

// The InvokeModelWithResponseStream API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
meta.html
var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in Llama 2's instruction format.
var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
prompt);

// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/generation").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    })
    )
    .build();
```

```
        }).build()).build());

    try {
        // Send the request and wait for the handler to process the response.
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- API の詳細については、「API リファレンス [InvokeModelWithResponseStream](#)」の「」を参照してください。AWS SDK for Java 2.x

### InvokeModelWithResponseStream: ラマ 3

次のコード例は、モデル呼び出し API を使用して Meta Llama 3 にテキストメッセージを送信し、レスポンスストリームを出力する方法を示しています。

#### SDK for Java 2.x

##### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

モデル呼び出し API を使用してテキストメッセージを送信し、レスポンスストリームをリアルタイムで処理します。

```
// Use the native inference API to send a text message to Meta Llama 3
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Llama3_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        meta.html
```

```
var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in Llama 3's instruction format.
var instruction = (
    "<|begin_of_text|>\n" +
    "<|start_header_id|>user<|end_header_id|>\n" +
    "{{prompt}} <|eot_id|>\n" +
    "<|start_header_id|>assistant<|end_header_id|>\n"
).replace("{{prompt}}", prompt);

// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONObject("/generation").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

try {
    // Send the request and wait for the handler to process the response.
```

```
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- API の詳細については、「API リファレンス [InvokeModelWithResponseStream](#)」の「」を参照してください。AWS SDK for Java 2.x

## ミスラル AI

### 会話

次のコード例は、Bedrock の Converse API を使用して Mistral にテキストメッセージを送信する方法を示しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

Bedrock の Converse API を使用して、Mistral にテキストメッセージを送信します。

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

非同期 Java クライアントで Bedrock の Converse API を使用して、Mistral にテキストメッセージを送信します。

```
// Use the Converse API to send a text message to Mistral
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
```

```
var client = BedrockRuntimeAsyncClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Mistral Large.
var modelId = "mistral.mistral-large-2402-v1:0";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
```

```
        // Wait for the future object to complete and retrieve the generated
text.
        String responseText = future.get();
        System.out.println(responseText);

        return responseText;

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converseAsync();
}
}
```

- API の詳細については、「[API リファレンス](#)」の AWS SDK for Java 2.x 「会話」を参照してください。

## ConverseStream

次のコード例は、Bedrock の Converse API を使用して Mistral にテキストメッセージを送信し、レスポンスストリームをリアルタイムで処理する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

Bedrock の Converse API を使用して Mistral にテキストメッセージを送信し、レスポンスストリームをリアルタイムで処理します。

```
// Use the Converse API to send a text message to Mistral
// and print the response stream.
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
            .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                .onContentBlockDelta(chunk -> {
                    String responseText = chunk.delta().text();
                    System.out.print(responseText);
                }).build()
            ).onError(err ->
```

```
        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
            .build());

    try {
        // Send the message with a basic inference configuration and attach the
handler.
        client.converseStream(request -> request.modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)
            ), responseStreamHandler).get();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    }
}
}
```

- API の詳細については、「API リファレンス [ConverseStream](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## InvokeModel

次のコード例は、Invoke Model API を使用して Mistral モデルにテキストメッセージを送信する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

モデル呼び出し API を使用してテキストメッセージを送信します。

```
// Use the native inference API to send a text message to Mistral.
```

```
import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        mistral-text-completion.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Mistral's instruction format.
        var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
        prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
        instruction);

        try {
            // Encode and send the request to the Bedrock Runtime.
```

```
        var response = client.invokeModel(request -> request
            .body(SdkBytes.fromUtf8String(nativeRequest))
            .modelId(modelId)
        );

        // Decode the response body.
        var responseBody = new JSONObject(response.body().asUtf8String());

        // Retrieve the generated text from the model's response.
        var text = new JSONPointer("/outputs/0/
text").queryFrom(responseBody).toString();
        System.out.println(text);

        return text;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    invokeModel();
}
}
```

- API の詳細については、「API リファレンス [InvokeModel](#)」の「」を参照してください。AWS SDK for Java 2.x

## InvokeModelWithResponseStream

次のコード例は、Invoke Model API を使用して Mistral AI モデルにテキストメッセージを送信し、レスポンスストリームを出力する方法を示しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

モデル呼び出し API を使用してテキストメッセージを送信し、レスポンスストリームをリアルタイムで処理します。

```
// Use the native inference API to send a text message to Mistral
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();
```



```
// Set the model ID, e.g., Mistral Large.
var modelId = "mistral.mistral-large-2402-v1:0";

// The InvokeModelWithResponseStream API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
mistral-text-completion.html
var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in Mistral's instruction format.
var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
prompt);

// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/outputs/0/
text").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
```

```
        completeResponseTextBuffer.append(text);
    }).build()).build();

    try {
        // Send the request and wait for the handler to process the response.
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- API の詳細については、「API リファレンス [InvokeModelWithResponseStream](#)」の「」を参照してください。AWS SDK for Java 2.x

## シナリオ

Amazon Bedrock 基盤モデルとやりとりするためのプレイグラウンドアプリケーションを作成します。

次のコード例は、さまざまな方法で Amazon Bedrock 基盤モデルと相互作用するプレイグラウンドを作成する方法を示しています。

### SDK for Java 2.x

Java 基盤モデル (FM) プレイグラウンドは Spring Boot のサンプルアプリケーションで、Java で Amazon Bedrock を使用する方法を紹介しています。この例は、Java 開発者が Amazon Bedrock を使用して生成系 AI 対応アプリケーションを構築する方法を示しています。次の 3 つのプレイグラウンドを使用して Amazon Bedrock 基盤モデルをテストしたり操作したりできます。

- テキストプレイグラウンド。
- チャットプレイグラウンド。
- イメージプレイグラウンド。

この例には、アクセスできる基盤モデルとその特性が一覧表示されています。ソースコードとデプロイ手順については、「」の「プロジェクト」を参照してください[GitHub](#)。

この例で使用されているサービス

- Amazon Bedrock ランタイム

## Stable Diffusion

### InvokeModel

次のコード例は、Amazon Bedrock で Stability.ai Stable Diffusion XL を呼び出してイメージを生成する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

Stable Diffusion を使用してイメージを作成します。

```
// Create an image with Stable Diffusion.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;
```

```
public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Stable Diffusion XL v1.
        var modelId = "stability.stable-diffusion-xl-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        diffusion-1-0-text-image.html
        var nativeRequestTemplate = ""
            {
                "text_prompts": [{ "text": "{{prompt}}" }],
                "style_preset": "{{style}}",
                "seed": {{seed}}
            }"";

        // Define the prompt for the image generation.
        var prompt = "A stylized picture of a cute old steampunk robot";

        // Get a random 32-bit seed for the image generation (max. 4,294,967,295).
        var seed = new BigInteger(31, new SecureRandom());

        // Choose a style preset.
        var style = "cinematic";

        // Embed the prompt, seed, and style in the model's native request payload.
        String nativeRequest = nativeRequestTemplate
            .replace("{{prompt}}", prompt)
            .replace("{{seed}}", seed.toString())
            .replace("{{style}}", style);

        try {
```

```
// Encode and send the request to the Bedrock Runtime.
var response = client.invokeModel(request -> request
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
);

// Decode the response body.
var responseBody = new JSONObject(response.body().asUtf8String());

// Retrieve the generated image data from the model's response.
var base64ImageData = new JSONPointer("/artifacts/0/base64")
    .queryFrom(responseBody)
    .toString();

return base64ImageData;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}
}

public static void main(String[] args) {
    System.out.println("Generating image. This may take a few seconds...");

    String base64ImageData = invokeModel();

    displayImage(base64ImageData);
}
}
```

- API の詳細については、「API リファレンス [InvokeModel](#)」の「」を参照してください。AWS SDK for Java 2.x

## CloudFront SDK for Java 2.x を使用した の例

次のコード例は、AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています CloudFront。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- [アクション](#)
- [シナリオ](#)

アクション

## CreateDistribution

次の例は、CreateDistribution を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

次の例では、Amazon Simple Storage Service (Amazon S3) バケットをコンテンツオリジンとして使用しています。

ディストリビューションを作成した後、コードはディストリビューションがデプロイされるまで待機 [CloudFrontWaiter](#) する を作成します。

```
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;
    }
}
```

- API の詳細については、「API リファレンス [CreateDistribution](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateFunction

次の例は、CreateFunction を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例  
を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionRequest;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionResponse;
import software.amazon.awssdk.services.cloudfront.model.FunctionConfig;
import software.amazon.awssdk.services.cloudfront.model.FunctionRuntime;
import java.io.InputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateFunction {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <functionName> <filePath>

            Where:
                functionName - The name of the function to create.\s
                filePath - The path to a file that contains the application
            logic for the function.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
            .region(Region.AWS_GLOBAL)
            .build();
```

```
String funArn = createNewFunction(cloudFrontClient, functionName, filePath);
System.out.println("The function ARN is " + funArn);
cloudFrontClient.close();
}

public static String createNewFunction(CloudFrontClient cloudFrontClient, String
functionName, String filePath) {
    try {
        InputStream fileIs =
CreateFunction.class.getClassLoader().getResourceAsStream(filePath);
        SdkBytes functionCode = SdkBytes.fromInputStream(fileIs);

        FunctionConfig config = FunctionConfig.builder()
            .comment("Created by using the CloudFront Java API")
            .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .name(functionName)
            .functionCode(functionCode)
            .functionConfig(config)
            .build();

        CreateFunctionResponse response =
cloudFrontClient.createFunction(functionRequest);
        return response.functionSummary().functionMetadata().functionARN();

    } catch (CloudFrontException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
}
```

- APIの詳細については、「APIリファレンス[CreateFunction](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateKeyGroup

次の例は、CreateKeyGroupを使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

キーグループには、署名付き URL または Cookie の検証に使用されるパブリックキーが少なくとも 1 つ必要です。

```
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;
    }
}
```

- API の詳細については、「API リファレンス [CreateKeyGroup](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreatePublicKey

次の例は、CreatePublicKey を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

次のコード例では、パブリックキーを読み取り、Amazon にアップロードします 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);
        }
    }
}
```

- API の詳細については、「API リファレンス [CreatePublicKey](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteDistribution

次の例は、DeleteDistribution を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

次のコード例では、配信を無効状態に更新し、ウェイターを使用して変更がデプロイされるのを待ってから、ディストリビューションを削除します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;

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

    public static void deleteDistribution(final CloudFrontClient
        cloudFrontClient, final String distributionId) {
        // First, disable the distribution by updating it.
        GetDistributionResponse response =
            cloudFrontClient.getDistribution(b -> b
                .id(distributionId));
        String etag = response.eTag();
        DistributionConfig distConfig =
            response.distribution().distributionConfig();
```

```

        cloudFrontClient.updateDistribution(builder -> builder
            .id(distributionId)
            .distributionConfig(builder1 -> builder1

                .cacheBehaviors(distConfig.cacheBehaviors())

                .defaultCacheBehavior(distConfig.defaultCacheBehavior())
                    .enabled(false)
                    .origins(distConfig.origins())
                    .comment(distConfig.comment())

                .callerReference(distConfig.callerReference())

                .defaultCacheBehavior(distConfig.defaultCacheBehavior())
                    .priceClass(distConfig.priceClass())
                    .aliases(distConfig.aliases())
                    .logging(distConfig.logging())

                .defaultRootObject(distConfig.defaultRootObject())

                .customErrorResponses(distConfig.customErrorResponses())

                .httpVersion(distConfig.httpVersion())

                .isIPV6Enabled(distConfig.isIPV6Enabled())

                .restrictions(distConfig.restrictions())

                .viewerCertificate(distConfig.viewerCertificate())
                    .webACLId(distConfig.webACLId())

                .originGroups(distConfig.originGroups())
                    .ifMatch(etag));

        logger.info("Distribution [{}] is DISABLED, waiting for deployment
before deleting ...",
            distributionId);
        GetDistributionResponse distributionResponse;
        try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse>
responseOrException = cfWaiter

```



```
                .waitUntilDistributionDeployed(builder ->
builder.id(distributionId)).matched();
                distributionResponse = responseOrException.response()
                .orElseThrow(() -> new
RuntimeException("Could not disable distribution"));
            }

            DeleteDistributionResponse deleteDistributionResponse =
cloudFrontClient
                .deleteDistribution(builder -> builder
                .id(distributionId)

                .ifMatch(distributionResponse.eTag()));
            if (deleteDistributionResponse.sdkHttpResponse().isSuccessful()) {
                logger.info("Distribution [{}] DELETED", distributionId);
            }
        }
    }
}
```

- API の詳細については、「API リファレンス [DeleteDistribution](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## UpdateDistribution

次の例は、UpdateDistribution を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [UpdateDistribution](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## シナリオ

### 署名リソースを削除する

次のコード例は、Amazon Simple Storage Service (Amazon S3) バケット内の制限付きコンテンツへのアクセス権を取得するために使用されるリソースを削除する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteKeyGroupResponse;
import
    software.amazon.awssdk.services.cloudfront.model.DeleteOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.DeletePublicKeyResponse;
import software.amazon.awssdk.services.cloudfront.model.GetKeyGroupResponse;
import
    software.amazon.awssdk.services.cloudfront.model.GetOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.GetPublicKeyResponse;

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

    public static void deleteOriginAccessControl(final CloudFrontClient
        cloudFrontClient,
        final String originAccessControlId) {
        GetOriginAccessControlResponse getResponse = cloudFrontClient
            .getOriginAccessControl(b -> b.id(originAccessControlId));
        DeleteOriginAccessControlResponse deleteResponse =
            cloudFrontClient.deleteOriginAccessControl(builder -> builder
```

```
        .id(originAccessControlId)
        .ifMatch(getResponse.eTag()));
    if (deleteResponse.sdkHttpResponse().isSuccessful()) {
        logger.info("Successfully deleted Origin Access Control [{}]",
originAccessControlId);
    }
}

    public static void deleteKeyGroup(final CloudFrontClient cloudFrontClient, final
String keyGroupId) {

        GetKeyGroupResponse getResponse = cloudFrontClient.getKeyGroup(b ->
b.id(keyGroupId));
        DeleteKeyGroupResponse deleteResponse =
cloudFrontClient.deleteKeyGroup(builder -> builder
            .id(keyGroupId)
            .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Key Group [{}]", keyGroupId);
        }
    }

    public static void deletePublicKey(final CloudFrontClient cloudFrontClient,
final String publicKeyId) {
        GetPublicKeyResponse getResponse = cloudFrontClient.getPublicKey(b ->
b.id(publicKeyId));

        DeletePublicKeyResponse deleteResponse =
cloudFrontClient.deletePublicKey(builder -> builder
            .id(publicKeyId)
            .ifMatch(getResponse.eTag()));

        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Public Key [{}]", publicKeyId);
        }
    }
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [DeleteKeyGroup](#)

- [DeleteOriginAccessControl](#)
- [DeletePublicKey](#)

## URL および cookies に署名する

次のコード例は、制限付きリソースへのアクセスを許可する署名付き URL と Cookie を作成する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

[CannedSignerRequest](#) クラスを使用して、既定ポリシーで URLs または Cookie に署名します。

```
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCannedPolicyRequest {

    public static CannedSignerRequest createRequestForCannedPolicy(String
distributionDomainName,
        String fileNameToUpload,
        String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
resourcePath).toString();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CannedSignerRequest.builder()
```

```
        .resourceUrl(cloudFrontUrl)
        .privateKey(path)
        .keyPairId(publicKeyId)
        .expirationDate(expirationDate)
        .build();
    }
}
```

[CustomSignerRequest](#) クラスを使用して、カスタムポリシーで URLs または Cookie に署名します。 `activeDate` および `ipRange` はオプションのメソッドです。

```
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCustomPolicyRequest {

    public static CustomSignerRequest createRequestForCustomPolicy(String
distributionDomainName,
        String fileNameToUpload,
        String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
resourcePath).toString();
        Instant expireDate = Instant.now().plus(7, ChronoUnit.DAYS);
        // URL will be accessible tomorrow using the signed URL.
        Instant activeDate = Instant.now().plus(1, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CustomSignerRequest.builder()
            .resourceUrl(cloudFrontUrl)
            .privateKey(path)
            .keyPairId(publicKeyId)
            .expirationDate(expireDate)
            .activeDate(activeDate) // Optional.
            // .ipRange("192.168.0.1/24") // Optional.
    }
}
```

```
        .build();
    }
}
```

次の例は、[CloudFrontUtilities](#) クラスを使用して署名付き Cookie と URLs を生成する方法を示しています。でこのコード例 [を表示します](#) 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;
    }
}
```

- API の詳細については、「API リファレンス [CloudFrontUtilities](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CloudWatch SDK for Java 2.x を使用した の例

次のコード例は、AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています CloudWatch。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

## 開始方法

こんにちは CloudWatchは

次のコード例は、 の使用を開始する方法を示しています CloudWatch。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloService {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <namespace>\s

                Where:
                namespace - The namespace to filter against (for example, AWS/
EC2).\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String namespace = args[0];
    Region region = Region.US_EAST_1;
    CloudWatchClient cw = CloudWatchClient.builder()
        .region(region)
        .build();

    listMets(cw, namespace);
    cw.close();
}

public static void listMets(CloudWatchClient cw, String namespace) {
    try {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> System.out.println(" Retrieved metric is: "
+ metrics.metricName()));

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

- API の詳細については、「API リファレンス [ListMetrics](#)」の「」を参照してください。AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### DeleteAlarms

次の例は、DeleteAlarms を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteAlarm {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <alarmName>

                Where:
                alarmName - An alarm name to delete (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String alarmName = args[0];
Region region = Region.US_EAST_2;
CloudWatchClient cw = CloudWatchClient.builder()
    .region(region)
    .build();

deleteCWAlarm(cw, alarmName);
cw.close();
}

public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.printf("Successfully deleted alarm %s", alarmName);

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

- API の詳細については、「API リファレンス [DeleteAlarms](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteAnomalyDetector

次の例は、DeleteAnomalyDetector を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .stat("Maximum")
            .build();

        DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
            .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
            .build();

        cw.deleteAnomalyDetector(request);
        System.out.println("Successfully deleted the Anomaly Detector.");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

- API の詳細については、「API リファレンス [DeleteAnomalyDetector](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteDashboards

次の例は、DeleteDashboards を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
    try {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();
        cw.deleteDashboards(dashboardsRequest);
        System.out.println(dashboardName + " was successfully deleted.");

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DeleteDashboards](#)」の「」を参照してください。  
AWS SDK for Java 2.x

**DescribeAlarmHistory**

次の例は、DescribeAlarmHistory を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String alarmName = rootNode.findValue("exampleAlarmName").asText();

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();
        DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
            .startDate(start)
            .endDate(endDate)
            .alarmName(alarmName)
            .historyItemType(HistoryItemType.ACTION)
            .build();

        DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
        List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
        if (historyItems.isEmpty()) {
            System.out.println("No alarm history data found for " + alarmName +
".");
        } else {
            for (AlarmHistoryItem item : historyItems) {
                System.out.println("History summary: " + item.historySummary());
                System.out.println("Time stamp: " + item.timestamp());
            }
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- APIの詳細については、「APIリファレンス[DescribeAlarmHistory](#)」の「」を参照してください。AWS SDK for Java 2.x



## DescribeAlarms

次の例は、DescribeAlarms を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void describeAlarms(CloudWatchClient cw) {
    try {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);

        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
        List<MetricAlarm> alarmList = response.metricAlarms();
        for (MetricAlarm alarm : alarmList) {
            System.out.println("Alarm name: " + alarm.alarmName());
            System.out.println("Alarm description: " +
alarm.alarmDescription());
        }
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DescribeAlarms](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DescribeAlarmsForMetric

次の例は、DescribeAlarmsForMetric を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        boolean hasAlarm = false;
        int retries = 10;

        DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        while (!hasAlarm && retries > 0) {
            DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
            hasAlarm = response.hasMetricAlarms();
            retries--;
            Thread.sleep(20000);
            System.out.println(".");
        }
        if (!hasAlarm)
            System.out.println("No Alarm state found for " + customMetricName +
" after 10 retries.");
        else
```

```
        System.out.println("Alarm state found for " + customMetricName +
        ".");
    } catch (CloudWatchException | IOException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DescribeAlarmsForMetric](#)」の「」を参照してください。AWS SDK for Java 2.x

## DescribeAnomalyDetectors

次の例は、DescribeAnomalyDetectors を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
            .maxResults(10)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
```

```
        .build();

        DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
        List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
        for (AnomalyDetector detector : anomalyDetectorList) {
            System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
            System.out.println("State: " + detector.stateValue());
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DescribeAnomalyDetectors](#)」の「」を参照してください。AWS SDK for Java 2.x

## DisableAlarmActions

次の例は、DisableAlarmActions を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DisableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DisableAlarmActions {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <alarmName>

            Where:
                alarmName - An alarm name to disable (for example, MyAlarm).
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
            .region(region)
            .build();

        disableActions(cw, alarmName);
        cw.close();
    }

    public static void disableActions(CloudWatchClient cw, String alarmName) {
        try {
            DisableAlarmActionsRequest request =
            DisableAlarmActionsRequest.builder()
                .alarmNames(alarmName)
                .build();

            cw.disableAlarmActions(request);
            System.out.printf("Successfully disabled actions on alarm %s",
            alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

```
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [DisableAlarmActions](#)」の「」を参照してください。AWS SDK for Java 2.x

## EnableAlarmActions

次の例は、EnableAlarmActions を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnableAlarmActions {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
        <alarmName>
```

```
        Where:
            alarmName - An alarm name to enable (for example, MyAlarm).
            """";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String alarm = args[0];
    Region region = Region.US_EAST_1;
    CloudWatchClient cw = CloudWatchClient.builder()
        .region(region)
        .build();

    enableActions(cw, alarm);
    cw.close();
}

public static void enableActions(CloudWatchClient cw, String alarm) {
    try {
        EnableAlarmActionsRequest request = EnableAlarmActionsRequest.builder()
            .alarmNames(alarm)
            .build();

        cw.enableAlarmActions(request);
        System.out.printf("Successfully enabled actions on alarm %s", alarm);

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

- APIの詳細については、「APIリファレンス[EnableAlarmActions](#)」の「」を参照してください。AWS SDK for Java 2.x

## GetMetricData

次の例は、GetMetricData を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set the date.
        Instant nowDate = Instant.now();

        long hours = 1;
        long minutes = 30;
        Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
ChronoUnit.MINUTES);

        Metric met = Metric.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        MetricStat metStat = MetricStat.builder()
            .stat("Maximum")
            .period(1)
            .metric(met)
            .build();

        MetricDataQuery dataQuery = MetricDataQuery.builder()
            .metricStat(metStat)
            .id("foo2")
            .returnData(true)
```



```
        .build();

List<MetricDataQuery> dq = new ArrayList<>();
dq.add(dataQuery);

GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
    .maxDatapoints(10)
    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
    .startTime(nowDate)
    .endTime(date2)
    .metricDataQueries(dq)
    .build();

GetMetricDataResponse response = cw.getMetricData(getMetReq);
List<MetricDataResult> data = response.metricDataResults();
for (MetricDataResult item : data) {
    System.out.println("The label is " + item.label());
    System.out.println("The status code is " +
item.statusCode().toString());
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス [GetMetricData](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## GetMetricStatistics

次の例は、`GetMetricStatistics` を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
        String metricOption, String date, Dimension myDimension) {
    try {
        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
        .endTime(endDate)
        .startTime(start)
        .dimensions(myDimension)
        .metricName(metVal)
        .namespace(nameSpace)
        .period(86400)
        .statistics(Statistic.fromValue(metricOption))
        .build();

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
                System.out
                    .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
            }
        } else {
            System.out.println("The returned data list is empty");
        }

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス[GetMetricStatistics](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## GetMetricWidgetImage

次の例は、GetMetricWidgetImage を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
    System.out.println("Getting Image data for custom metric.");
    try {
        String myJSON = "{\n" +
            "  \"title\": \"Example Metric Graph\",\n" +
            "  \"view\": \"timeSeries\",\n" +
            "  \"stacked\": false,\n" +
            "  \"period\": 10,\n" +
            "  \"width\": 1400,\n" +
            "  \"height\": 600,\n" +
            "  \"metrics\": [\n" +
            "    [\n" +
            "      \"AWS/Billing\",\n" +
            "      \"EstimatedCharges\",\n" +
            "      \"Currency\",\n" +
            "      \"USD\"\n" +
            "    ]\n" +
            "  ]\n" +
            "}";

        GetMetricWidgetImageRequest imageRequest =
        GetMetricWidgetImageRequest.builder()
            .metricWidget(myJSON)
            .build();

        GetMetricWidgetImageResponse response =
        cw.getMetricWidgetImage(imageRequest);
        SdkBytes sdkBytes = response.metricWidgetImage();
        byte[] bytes = sdkBytes.asByteArray();
        File outputFile = new File(fileName);
        try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
```

```
        outputStream.write(bytes);
    }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [GetMetricWidgetImage](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListDashboards

次の例は、ListDashboards を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void listDashboards(CloudWatchClient cw) {
    try {
        ListDashboardsIterable listRes = cw.listDashboardsPaginator();
        listRes.stream()
            .flatMap(r -> r.dashboardEntries().stream())
            .forEach(entry -> {
                System.out.println("Dashboard name is: " +
entry.dashboardName());
                System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
            });

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

```
}
```

- API の詳細については、「API リファレンス [ListDashboards](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListMetrics

次の例は、ListMetrics を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListMetrics {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <namespace>\s

                Where:
```

```
namespace - The namespace to filter against (for example, AWS/
EC2).\s
    """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String namespace = args[0];
Region region = Region.US_EAST_1;
CloudWatchClient cw = CloudWatchClient.builder()
    .region(region)
    .build();

listMets(cw, namespace);
cw.close();
}

public static void listMets(CloudWatchClient cw, String namespace) {
    boolean done = false;
    String nextToken = null;

    try {
        while (!done) {

            ListMetricsResponse response;
            if (nextToken == null) {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

                response = cw.listMetrics(request);
            } else {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .nextToken(nextToken)
                    .build();

                response = cw.listMetrics(request);
            }

            for (Metric metric : response.metrics()) {
                System.out.printf("Retrieved metric %s", metric.metricName());
            }
        }
    }
}
```

```
        System.out.println();
    }

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

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

- API の詳細については、「API リファレンス [ListMetrics](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutAnomalyDetector

次の例は、PutAnomalyDetector を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
```

```
String customMetricName =
rootNode.findValue("customMetricName").asText();

SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
    .metricName(customMetricName)
    .namespace(customMetricNamespace)
    .stat("Maximum")
    .build();

PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
    .build();

cw.putAnomalyDetector(anomalyDetectorRequest);
System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- APIの詳細については、「APIリファレンス[PutAnomalyDetector](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutDashboard

次の例は、PutDashboardを使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。



```
public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();

        PutDashboardResponse response = cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully created.");
        List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
        if (messages.isEmpty()) {
            System.out.println("There are no messages in the new Dashboard");
        } else {
            for (DashboardValidationMessage message : messages) {
                System.out.println("Message is: " + message.message());
            }
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [PutDashboard](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## PutMetricAlarm

次の例は、PutMetricAlarm を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String createAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();

        cw.putMetricAlarm(alarmRequest);
        System.out.println(alarmName + " was successfully created!");
        return alarmName;
    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
        return "";  
    }
```

- API の詳細については、「API リファレンス [PutMetricAlarm](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## PutMetricData

次の例は、PutMetricData を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {  
    try {  
        // Read values from the JSON file.  
        JsonParser parser = new JsonFactory().createParser(new File(fileName));  
        com.fasterxml.jackson.databind.JsonNode rootNode = new  
ObjectMapper().readTree(parser);  
        String customMetricNamespace =  
rootNode.findValue("customMetricNamespace").asText();  
        String customMetricName =  
rootNode.findValue("customMetricName").asText();  
  
        // Set an Instant object.  
        String time =  
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);  
        Instant instant = Instant.parse(time);  
  
        MetricDatum datum = MetricDatum.builder()  
            .metricName(customMetricName)  
            .unit(StandardUnit.NONE)  
            .value(1001.00)  
            .timestamp(instant)  
            .build();
```

```
        MetricDatum datum2 = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1002.00)
            .timestamp(instant)
            .build();

        List<MetricDatum> metricDataList = new ArrayList<>();
        metricDataList.add(datum);
        metricDataList.add(datum2);

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace(customMetricNamespace)
            .metricData(metricDataList)
            .build();

        cw.putMetricData(request);
        System.out.println("Added metric values for for metric " +
            customMetricName);

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [PutMetricData](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## シナリオ

メトリクス、ダッシュボード、およびアラームの使用を開始する

次のコードサンプルは、以下の操作方法を示しています。

- CloudWatch 名前空間とメトリクスを一覧表示します。
- メトリクスと予想請求額の統計の取得
- ダッシュボードの作成と更新
- メトリクスの作成とデータの追加

- アラームの作成/トリガーとアラーム履歴の表示
- 異常ディテクターの追加
- メトリクス画像の取得とリソースのクリーンアップ

## SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.AlarmHistoryItem;
import software.amazon.awssdk.services.cloudwatch.model.AlarmType;
import software.amazon.awssdk.services.cloudwatch.model.AnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import software.amazon.awssdk.services.cloudwatch.model.DashboardValidationMessage;
import software.amazon.awssdk.services.cloudwatch.model.Datapoint;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryResponse;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricResponse;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsRequest;
```

```
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataResponse;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageResponse;
import software.amazon.awssdk.services.cloudwatch.model.HistoryItemType;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataQuery;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataResult;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.MetricStat;
import software.amazon.awssdk.services.cloudwatch.model.PutAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.ScanBy;
import software.amazon.awssdk.services.cloudwatch.model.SingleMetricAnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import software.amazon.awssdk.services.cloudwatch.paginators.ListDashboardsIterable;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;
```

```
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To enable billing metrics and statistics for this example, make sure billing
 * alerts are enabled for your account:
 * https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/monitor\_estimated\_charges\_with\_cloudwatch.html#turning\_on\_billing\_metrics
 *
 * This Java code example performs the following tasks:
 *
 * 1. List available namespaces from Amazon CloudWatch.
 * 2. List available metrics within the selected Namespace.
 * 3. Get statistics for the selected metric over the last day.
 * 4. Get CloudWatch estimated billing for the last week.
 * 5. Create a new CloudWatch dashboard with metrics.
 * 6. List dashboards using a paginator.
 * 7. Create a new custom metric by adding data for it.
 * 8. Add the custom metric to the dashboard.
 * 9. Create an alarm for the custom metric.
 * 10. Describe current alarms.
 * 11. Get current data for the new custom metric.
 * 12. Push data into the custom metric to trigger the alarm.
 * 13. Check the alarm state using the action DescribeAlarmsForMetric.
 * 14. Get alarm history for the new alarm.
 * 15. Add an anomaly detector for the custom metric.
 * 16. Describe current anomaly detectors.
 * 17. Get a metric image for the custom metric.
 * 18. Clean up the Amazon CloudWatch resources.
 */
public class CloudWatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = ""

                Usage:
```

```

    <myDate> <costDateWeek> <dashboardName> <dashboardJson>
<dashboardAdd> <settings> <metricImage> \s

```

Where:

myDate - The start date to use to get metric statistics. (For example, 2023-01-11T18:35:24.00Z.)\s

costDateWeek - The start date to use to get AWS/Billinget statistics. (For example, 2023-01-11T18:35:24.00Z.)\s

dashboardName - The name of the dashboard to create.\s

dashboardJson - The location of a JSON file to use to create a dashboard. (See Readme file.)\s

dashboardAdd - The location of a JSON file to use to update a dashboard. (See Readme file.)\s

settings - The location of a JSON file from which various values are read. (See Readme file.)\s

metricImage - The location of a BMP file that is used to create a graph.\s

```

    """;

```

```

if (args.length != 7) {
    System.out.println(usage);
    System.exit(1);
}

```

```

Region region = Region.US_EAST_1;
String myDate = args[0];
String costDateWeek = args[1];
String dashboardName = args[2];
String dashboardJson = args[3];
String dashboardAdd = args[4];
String settings = args[5];
String metricImage = args[6];

```

```

Double dataPoint = Double.parseDouble("10.0");
Scanner sc = new Scanner(System.in);
CloudWatchClient cw = CloudWatchClient.builder()
    .region(region)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

```

```

System.out.println(DASHES);
System.out.println("Welcome to the Amazon CloudWatch example scenario.");
System.out.println(DASHES);

```



```
System.out.println(DASHES);
System.out.println(
    "1. List at least five available unique namespaces from Amazon
CloudWatch. Select one from the list.");
ArrayList<String> list = listNameSpaces(cw);
for (int z = 0; z < 5; z++) {
    int index = z + 1;
    System.out.println("    " + index + ". " + list.get(z));
}

String selectedNamespace = "";
String selectedMetrics = "";
int num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    selectedNamespace = list.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + selectedNamespace);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. List available metrics within the selected namespace
and select one from the list.");
ArrayList<String> metList = listMets(cw, selectedNamespace);
for (int z = 0; z < 5; z++) {
    int index = z + 1;
    System.out.println("    " + index + ". " + metList.get(z));
}
num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    selectedMetrics = metList.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + selectedMetrics);
Dimension myDimension = getSpecificMet(cw, selectedNamespace);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get statistics for the selected metric over the last
day.");
```

```
String metricOption = "";
ArrayList<String> statTypes = new ArrayList<>();
statTypes.add("SampleCount");
statTypes.add("Average");
statTypes.add("Sum");
statTypes.add("Minimum");
statTypes.add("Maximum");

for (int t = 0; t < 5; t++) {
    System.out.println("    " + (t + 1) + ". " + statTypes.get(t));
}
System.out.println("Select a metric statistic by entering a number from the
preceding list:");
num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    metricOption = statTypes.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + metricOption);
getAndDisplayMetricStatistics(cw, selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get CloudWatch estimated billing for the last
week.");
getMetricStatistics(cw, costDateWeek);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Create a new CloudWatch dashboard with metrics.");
createDashboardWithMetrics(cw, dashboardName, dashboardJson);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. List dashboards using a paginator.");
listDashboards(cw);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Create a new custom metric by adding data to it.");
createNewCustomMetric(cw, dataPoint);
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Add an additional metric to the dashboard.");
addMetricToDashboard(cw, dashboardAdd, dashboardName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Create an alarm for the custom metric.");
String alarmName = createAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Describe ten current alarms.");
describeAlarms(cw);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Get current data for new custom metric.");
getCustomMetricData(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Push data into the custom metric to trigger the
alarm.");
addMetricDataForAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
checkForMetricAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Get alarm history for the new alarm.");
getAlarmHistory(cw, settings, myDate);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Add an anomaly detector for the custom metric.");
addAnomalyDetector(cw, settings);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("16. Describe current anomaly detectors.");
describeAnomalyDetectors(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("17. Get a metric image for the custom metric.");
getAndOpenMetricImage(cw, metricImage);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("18. Clean up the Amazon CloudWatch resources.");
deleteDashboard(cw, dashboardName);
deleteCWAlarm(cw, alarmName);
deleteAnomalyDetector(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The Amazon CloudWatch example scenario is complete.");
System.out.println(DASHES);
cw.close();
}

public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .stat("Maximum")
            .build();

        DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
            .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
```

```
        .build();

        cw.deleteAnomalyDetector(request);
        System.out.println("Successfully deleted the Anomaly Detector.");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.println("Successfully deleted alarm " + alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
    try {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();
        cw.deleteDashboards(dashboardsRequest);
        System.out.println(dashboardName + " was successfully deleted.");

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
```

```

System.out.println("Getting Image data for custom metric.");
try {
    String myJSON = "{\n" +
        "  \"title\": \"Example Metric Graph\",\n" +
        "  \"view\": \"timeSeries\",\n" +
        "  \"stacked\": false,\n" +
        "  \"period\": 10,\n" +
        "  \"width\": 1400,\n" +
        "  \"height\": 600,\n" +
        "  \"metrics\": [\n" +
        "    [\n" +
        "      \"AWS/Billing\",\n" +
        "      \"EstimatedCharges\",\n" +
        "      \"Currency\",\n" +
        "      \"USD\"\n" +
        "    ]\n" +
        "  ]\n" +
        "}";

    GetMetricWidgetImageRequest imageRequest =
    GetMetricWidgetImageRequest.builder()
        .metricWidget(myJSON)
        .build();

    GetMetricWidgetImageResponse response =
    cw.getMetricWidgetImage(imageRequest);
    SdkBytes sdkBytes = response.metricWidgetImage();
    byte[] bytes = sdkBytes.asByteArray();
    File outputFile = new File(fileName);
    try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
        outputStream.write(bytes);
    }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));

```

```
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
            .maxResults(10)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
        List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
        for (AnomalyDetector detector : anomalyDetectorList) {
            System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
            System.out.println("State: " + detector.stateValue());
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .stat("Maximum")
```

```
        .build();

        PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
        .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
        .build();

        cw.putAnomalyDetector(anomalyDetectorRequest);
        System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String alarmName = rootNode.findValue("exampleAlarmName").asText();

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();
        DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
        .startDate(start)
        .endDate(endDate)
        .alarmName(alarmName)
        .historyItemType(HistoryItemType.ACTION)
        .build();

        DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
        List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
        if (historyItems.isEmpty()) {
            System.out.println("No alarm history data found for " + alarmName +
".");
        } else {
            for (AlarmHistoryItem item : historyItems) {
```



```
        System.out.println("History summary: " + item.historySummary());
        System.out.println("Time stamp: " + item.timestamp());
    }
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        boolean hasAlarm = false;
        int retries = 10;

        DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        while (!hasAlarm && retries > 0) {
            DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
            hasAlarm = response.hasMetricAlarms();
            retries--;
            Thread.sleep(20000);
            System.out.println(".");
        }
        if (!hasAlarm)
            System.out.println("No Alarm state found for " + customMetricName +
" after 10 retries.");
        else
            System.out.println("Alarm state found for " + customMetricName +
".");
    }
}
```

```
    } catch (CloudWatchException | IOException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1001.00)
            .timestamp(instant)
            .build();

        MetricDatum datum2 = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1002.00)
            .timestamp(instant)
            .build();

        List<MetricDatum> metricDataList = new ArrayList<>();
        metricDataList.add(datum);
        metricDataList.add(datum2);

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace(customMetricNamespace)
            .metricData(metricDataList)
```

```
        .build();

        cw.putMetricData(request);
        System.out.println("Added metric values for for metric " +
customMetricName);

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set the date.
        Instant nowDate = Instant.now();

        long hours = 1;
        long minutes = 30;
        Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
ChronoUnit.MINUTES);

        Metric met = Metric.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        MetricStat metStat = MetricStat.builder()
            .stat("Maximum")
            .period(1)
            .metric(met)
            .build();

        MetricDataQuery dataQuery = MetricDataQuery.builder()
            .metricStat(metStat)
```

```
        .id("foo2")
        .returnData(true)
        .build();

List<MetricDataQuery> dq = new ArrayList<>();
dq.add(dataQuery);

GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
    .maxDatapoints(10)
    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
    .startTime(nowDate)
    .endTime(date2)
    .metricDataQueries(dq)
    .build();

GetMetricDataResponse response = cw.getMetricData(getMetReq);
List<MetricDataResult> data = response.metricDataResults();
for (MetricDataResult item : data) {
    System.out.println("The label is " + item.label());
    System.out.println("The status code is " +
item.statusCode().toString());
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void describeAlarms(CloudWatchClient cw) {
    try {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);

        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
        List<MetricAlarm> alarmList = response.metricAlarms();
        for (MetricAlarm alarm : alarmList) {
            System.out.println("Alarm name: " + alarm.alarmName());
        }
    }
}
```

```
        System.out.println("Alarm description: " +
alarm.alarmDescription());
    }
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String createAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

        .comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();
    }
}
```

```
        cw.putMetricAlarm(alarmRequest);
        System.out.println(alarmName + " was successfully created!");
        return alarmName;

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void addMetricToDashboard(CloudWatchClient cw, String fileName,
String dashboardName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();

        cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully updated.");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void createNewCustomMetric(CloudWatchClient cw, Double dataPoint)
{
    try {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
            .value("URLS")
            .build();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
```

```
        .value(dataPoint)
        .timestamp(instant)
        .dimensions(dimension)
        .build();

    PutMetricDataRequest request = PutMetricDataRequest.builder()
        .namespace("SITE/TRAFFIC")
        .metricData(datum)
        .build();

    cw.putMetricData(request);
    System.out.println("Added metric values for for metric PAGES_VISITED");

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void listDashboards(CloudWatchClient cw) {
    try {
        ListDashboardsIterable listRes = cw.listDashboardsPaginator();
        listRes.stream()
            .flatMap(r -> r.dashboardEntries().stream())
            .forEach(entry -> {
                System.out.println("Dashboard name is: " +
entry.dashboardName());
                System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
            });
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();
```

```
        PutDashboardResponse response = cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully created.");
        List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
        if (messages.isEmpty()) {
            System.out.println("There are no messages in the new Dashboard");
        } else {
            for (DashboardValidationMessage message : messages) {
                System.out.println("Message is: " + message.message());
            }
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String readFileAsString(String file) throws IOException {
    return new String(Files.readAllBytes(Paths.get(file)));
}

public static void getMetricStatistics(CloudWatchClient cw, String costDateWeek)
{
    try {
        Instant start = Instant.parse(costDateWeek);
        Instant endDate = Instant.now();
        Dimension dimension = Dimension.builder()
            .name("Currency")
            .value("USD")
            .build();

        List<Dimension> dimensionList = new ArrayList<>();
        dimensionList.add(dimension);
        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .metricName("EstimatedCharges")
            .namespace("AWS/Billing")
            .dimensions(dimensionList)
            .statistics(Statistic.MAXIMUM)
            .startTime(start)
            .endTime(endDate)
            .period(86400)
```



```
        .build();

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
                System.out
                    .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
            }
        } else {
            System.out.println("The returned data list is empty");
        }

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
        String metricOption, String date, Dimension myDimension) {
    try {
        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .endTime(endDate)
            .startTime(start)
            .dimensions(myDimension)
            .metricName(metVal)
            .namespace(nameSpace)
            .period(86400)
            .statistics(Statistic.fromValue(metricOption))
            .build();

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
```

```
        System.out
            .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
    }
    } else {
        System.out.println("The returned data list is empty");
    }

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static Dimension getSpecificMet(CloudWatchClient cw, String namespace) {
    try {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsResponse response = cw.listMetrics(request);
        List<Metric> myList = response.metrics();
        Metric metric = myList.get(0);
        return metric.dimensions().get(0);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static ArrayList<String> listMets(CloudWatchClient cw, String namespace)
{
    try {
        ArrayList<String> metList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> metList.add(metrics.metricName()));
    }
}
```

```
        return metList;

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static ArrayList<String> listNameSpaces(CloudWatchClient cw) {
    try {
        ArrayList<String> nameSpaceList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder()
            .build();

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> {
                String data = metrics.namespace();
                if (!nameSpaceList.contains(data)) {
                    nameSpaceList.add(data);
                }
            });

        return nameSpaceList;
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [DeleteAlarms](#)
  - [DeleteAnomalyDetector](#)
  - [DeleteDashboards](#)
  - [DescribeAlarmHistory](#)

- [DescribeAlarms](#)
- [DescribeAlarmsForMetric](#)
- [DescribeAnomalyDetectors](#)
- [GetMetricData](#)
- [GetMetricStatistics](#)
- [GetMetricWidgetImage](#)
- [ListMetrics](#)
- [PutAnomalyDetector](#)
- [PutDashboard](#)
- [PutMetricAlarm](#)
- [PutMetricData](#)

## CloudWatch SDK for Java 2.x を使用したイベントの例

次のコード例は、CloudWatch イベント AWS SDK for Java 2.x でを使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- [アクション](#)

## アクション

### PutEvents

次の例は、PutEvents を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutEvents {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <resourceArn>

                Where:
                resourceArn - An Amazon Resource Name (ARN) related to the
events.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String resourceArn = args[0];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();
```

```
        putCWEvents(cwe, resourceArn);
        cwe.close();
    }

    public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn) {
        try {
            final String EVENT_DETAILS = "{ \"key1\": \"value1\", \"key2\": \"value2\" }";

            PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
                .detail(EVENT_DETAILS)
                .detailType("sampleSubmitted")
                .resources(resourceArn)
                .source("aws-sdk-java-cloudwatch-example")
                .build();

            PutEventsRequest request = PutEventsRequest.builder()
                .entries(requestEntry)
                .build();

            cwe.putEvents(request);
            System.out.println("Successfully put CloudWatch event");

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

- API の詳細については、「API リファレンス[PutEvents](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutRule

次の例は、PutRule を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutRule {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <ruleName> roleArn\s

            Where:
                ruleName - A rule name (for example, myrule).
                roleArn - A role ARN value (for example,
arn:aws:iam::xxxxxx047983:user/MyUser).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String roleArn = args[1];
```

```
CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
    .build();

putCWRule(cwe, ruleName, roleArn);
cwe.close();
}

public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
roleArn) {
    try {
        PutRuleRequest request = PutRuleRequest.builder()
            .name(ruleName)
            .roleArn(roleArn)
            .scheduleExpression("rate(5 minutes)")
            .state(RuleState.ENABLED)
            .build();

        PutRuleResponse response = cwe.putRule(request);
        System.out.printf(
            "Successfully created CloudWatch events rule %s with arn %s",
            roleArn, response.ruleArn());

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

- APIの詳細については、「APIリファレンス[PutRule](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutTargets

次の例は、PutTargets を使用する方法を説明しています。



## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.Target;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutTargets {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <ruleName> <functionArn> <targetId>\s

            Where:
                ruleName - A rule name (for example, myrule).
                functionArn - An AWS Lambda function ARN (for example,
                arn:aws:lambda:us-west-2:xxxxxx047983:function:lamda1).
                targetId - A target id value.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String functionArn = args[1];
```

```
String targetId = args[2];
CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
    .build();

putCWTargets(cwe, ruleName, functionArn, targetId);
cwe.close();
}

public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName,
String functionArn, String targetId) {
    try {
        Target target = Target.builder()
            .arn(functionArn)
            .id(targetId)
            .build();

        PutTargetsRequest request = PutTargetsRequest.builder()
            .targets(target)
            .rule(ruleName)
            .build();

        cwe.putTargets(request);
        System.out.printf(
            "Successfully created CloudWatch events target for rule %s",
            ruleName);

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

- API の詳細については、「API リファレンス [PutTargets](#)」の「」を参照してください。AWS SDK for Java 2.x

## CloudWatch SDK for Java 2.x を使用したログの例

次のコード例は、を CloudWatch Logs AWS SDK for Java 2.x で使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック


- [アクション](#)

## アクション

### DeleteSubscriptionFilter

次の例は、DeleteSubscriptionFilter を使用する方法を説明しています。

SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DeleteSubscriptionFilterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteSubscriptionFilter {
```

```
public static void main(String[] args) {
    final String usage = ""

        Usage:
        <filter> <logGroup>

        Where:
        filter - The name of the subscription filter (for example,
MyFilter).
        logGroup - The name of the log group. (for example, testgroup).
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String filter = args[0];
    String logGroup = args[1];
    CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
        .build();

    deleteSubFilter(logs, filter, logGroup);
    logs.close();
}

public static void deleteSubFilter(CloudWatchLogsClient logs, String filter,
String logGroup) {
    try {
        DeleteSubscriptionFilterRequest request =
DeleteSubscriptionFilterRequest.builder()
            .filterName(filter)
            .logGroupName(logGroup)
            .build();

        logs.deleteSubscriptionFilter(request);
        System.out.printf("Successfully deleted CloudWatch logs subscription
filter %s", filter);

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

```
}
```

- API の詳細については、「API リファレンス [DeleteSubscriptionFilter](#)」の「」を参照してください。AWS SDK for Java 2.x

## DescribeSubscriptionFilters

次の例は、DescribeSubscriptionFilters を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersRequest;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.SubscriptionFilter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeSubscriptionFilters {
    public static void main(String[] args) {

        final String usage = ""

        Usage:
```

```
<logGroup>

Where:
  logGroup - A log group name (for example, myloggroup).
  """";

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String logGroup = args[0];
CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

describeFilters(logs, logGroup);
logs.close();
}

public static void describeFilters(CloudWatchLogsClient logs, String logGroup) {
    try {
        boolean done = false;
        String newToken = null;

        while (!done) {
            DescribeSubscriptionFiltersResponse response;
            if (newToken == null) {
                DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                    .logGroupName(logGroup)
                    .limit(1).build();

                response = logs.describeSubscriptionFilters(request);
            } else {
                DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                    .nextToken(newToken)
                    .logGroupName(logGroup)
                    .limit(1).build();
                response = logs.describeSubscriptionFilters(request);
            }

            for (SubscriptionFilter filter : response.subscriptionFilters()) {
```

```
        System.out.printf("Retrieved filter with name %s, " + "pattern
%s " + "and destination arn %s",
            filter.filterName(),
            filter.filterPattern(),
            filter.destinationArn());
    }

    if (response.nextToken() == null) {
        done = true;
    } else {
        newToken = response.nextToken();
    }
}

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

- API の詳細については、「API リファレンス [DescribeSubscriptionFilters](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutSubscriptionFilter

次の例は、PutSubscriptionFilter を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
```

```
import
software.amazon.awssdk.services.cloudwatchlogs.model.PutSubscriptionFilterRequest;

/**
 * Before running this code example, you need to grant permission to CloudWatch
 * Logs the right to execute your Lambda function.
 * To perform this task, you can use this CLI command:
 *
 * aws lambda add-permission --function-name "lamda1" --statement-id "lamda1"
 * --principal "logs.us-west-2.amazonaws.com" --action "lambda:InvokeFunction"
 * --source-arn "arn:aws:logs:us-west-2:111111111111:log-group:testgroup:*"
 * --source-account "111111111111"
 *
 * Make sure you replace the function name with your function name and replace
 * '111111111111' with your account details.
 * For more information, see "Subscription Filters with AWS Lambda" in the
 * Amazon CloudWatch Logs Guide.
 *
 * Also, before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <filter> <pattern> <logGroup> <functionArn>\s

            Where:
                filter - A filter name (for example, myfilter).
                pattern - A filter pattern (for example, ERROR).
                logGroup - A log group name (testgroup).
                functionArn - An AWS Lambda function ARN (for example,
arn:aws:lambda:us-west-2:111111111111:function:lambda1) .
                """;

        if (args.length != 4) {
```



```
        System.out.println(usage);
        System.exit(1);
    }

    String filter = args[0];
    String pattern = args[1];
    String logGroup = args[2];
    String functionArn = args[3];
    Region region = Region.US_WEST_2;
    CloudWatchLogsClient cwl = CloudWatchLogsClient.builder()
        .region(region)
        .build();

    putSubFilters(cwl, filter, pattern, logGroup, functionArn);
    cwl.close();
}

public static void putSubFilters(CloudWatchLogsClient cwl,
    String filter,
    String pattern,
    String logGroup,
    String functionArn) {

    try {
        PutSubscriptionFilterRequest request =
PutSubscriptionFilterRequest.builder()
            .filterName(filter)
            .filterPattern(pattern)
            .logGroupName(logGroup)
            .destinationArn(functionArn)
            .build();

        cwl.putSubscriptionFilter(request);
        System.out.printf(
            "Successfully created CloudWatch logs subscription filter %s",
            filter);

    } catch (CloudWatchLogsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [PutSubscriptionFilter](#)」の「」を参照してください。AWS SDK for Java 2.x

## StartLiveTail

次の例は、StartLiveTail を使用する方法を説明しています。

### SDK for Java 2.x

必要なファイルを含めます。

```
import io.reactivex.FlowableSubscriber;
import io.reactivex.annotations.NonNull;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsAsyncClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionStart;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionUpdate;
import software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailRequest;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseHandler;
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseStream;

import java.util.Date;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;
```

Live Tail セッションのイベントを処理します。

```
private static StartLiveTailResponseHandler
getStartLiveTailResponseStreamHandler(
    AtomicReference<Subscription> subscriptionAtomicReference) {
    return StartLiveTailResponseHandler.builder()
        .onResponse(r -> System.out.println("Received initial response"))
        .onError(throwable -> {
```

```
        CloudWatchLogsException e = (CloudWatchLogsException)
throwable.getCause();
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    })
    .subscriber(() -> new FlowableSubscriber<>() {
        @Override
        public void onSubscribe(@NonNull Subscription s) {
            subscriptionAtomicReference.set(s);
            s.request(Long.MAX_VALUE);
        }

        @Override
        public void onNext(StartLiveTailResponseStream event) {
            if (event instanceof LiveTailSessionStart) {
                LiveTailSessionStart sessionStart = (LiveTailSessionStart)
event;

                System.out.println(sessionStart);
            } else if (event instanceof LiveTailSessionUpdate) {
                LiveTailSessionUpdate sessionUpdate =
(LiveTailSessionUpdate) event;
                List<LiveTailSessionLogEvent> logEvents =
sessionUpdate.sessionResults();
                logEvents.forEach(e -> {
                    long timestamp = e.timestamp();
                    Date date = new Date(timestamp);
                    System.out.println "[" + date + "]" + e.message());
                });
            } else {
                throw CloudWatchLogsException.builder().message("Unknown
event type").build();
            }
        }

        @Override
        public void onError(Throwable throwable) {
            System.out.println(throwable.getMessage());
            System.exit(1);
        }

        @Override
        public void onComplete() {
            System.out.println("Completed Streaming Session");
        }
    })
}
```

```
    })
    .build();
}
```

Live Tail セッションを開始します。

```
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));
```

一定時間が経過したら Live Tail セッションを停止します。

```
/* Set a timeout for the session and cancel the subscription. This will:
 * 1). Close the stream
 * 2). Stop the Live Tail session
 */
try {
    Thread.sleep(10000);
} catch (InterruptedException e) {
    throw new RuntimeException(e);
}
if (subscriptionAtomicReference.get() != null) {
    subscriptionAtomicReference.get().cancel();
    System.out.println("Subscription to stream closed");
}
```

- API の詳細については、「API リファレンス [StartLiveTail](#)」の「」を参照してください。AWS SDK for Java 2.x

## SDK for Java 2.x を使用した Amazon Cognito ID の例

次のコード例は、Amazon Cognito Identity AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- [アクション](#)

### アクション

#### CreateIdentityPool

次の例は、CreateIdentityPool を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
```

```
import
software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import
software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <identityPoolName>\s

            Where:
                identityPoolName - The name to give your identity pool.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoolName = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String identityPoolId = createIdPool(cognitoClient, identityPoolName);
        System.out.println("Unity pool ID " + identityPoolId);
        cognitoClient.close();
    }

    public static String createIdPool(CognitoIdentityClient cognitoClient, String
identityPoolName) {
        try {
```

```
        CreateIdentityPoolRequest poolRequest =
CreateIdentityPoolRequest.builder()
    .allowUnauthenticatedIdentities(false)
    .identityPoolName(identityPoolName)
    .build();

        CreateIdentityPoolResponse response =
cognitoClient.createIdentityPool(poolRequest);
        return response.identityPoolId();

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- API の詳細については、「API リファレンス [CreateIdentityPool](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteIdentityPool

次の例は、DeleteIdentityPool を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteIdentityPool {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <identityPoolId>\s

            Where:
                identityPoolId - The Id value of your identity pool.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoolId = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        deleteIdPool(cognitoIdClient, identityPoolId);
        cognitoIdClient.close();
    }

    public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
identityPoolId) {
        try {

            DeleteIdentityPoolRequest identityPoolRequest =
DeleteIdentityPoolRequest.builder()
                .identityPoolId(identityPoolId)
                .build();
```



```
        cognitoIdClient.deleteIdentityPool(identityPoolRequest);
        System.out.println("Done");

    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [DeleteIdentityPool](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## GetCredentialsForIdentity

次の例は、GetCredentialsForIdentity を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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.CognitoIdentityProviderException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <identityId>\s

            Where:
                identityId - The Id of an existing identity in the format
REGION:GUID.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityId = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .build();

        getCredsForIdentity(cognitoClient, identityId);
        cognitoClient.close();
    }

    public static void getCredsForIdentity(CognitoIdentityClient cognitoClient,
String identityId) {
        try {
            GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
GetCredentialsForIdentityRequest
                .builder()
                .identityId(identityId)
                .build();

            GetCredentialsForIdentityResponse response = cognitoClient
                .getCredentialsForIdentity(getCredentialsForIdentityRequest);
            System.out.println(
                "Identity ID " + response.identityId() + ", Access key ID " +
response.credentials().accessKeyId());
        }
    }
}
```

```
        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [GetCredentialsForIdentity](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListIdentityPools

次の例は、ListIdentityPools を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
    software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
import
    software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentityPools {
```

```
public static void main(String[] args) {
    CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
        .region(Region.US_EAST_1)
        .build();

    listIdPools(cognitoClient);
    cognitoClient.close();
}

public static void listIdPools(CognitoIdentityClient cognitoClient) {
    try {
        ListIdentityPoolsRequest poolsRequest =
ListIdentityPoolsRequest.builder()
        .maxResults(15)
        .build();

        ListIdentityPoolsResponse response =
cognitoClient.listIdentityPools(poolsRequest);
        response.identityPools().forEach(pool -> {
            System.out.println("Pool ID: " + pool.identityPoolId());
            System.out.println("Pool name: " + pool.identityPoolName());
        });

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [ListIdentityPools](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## SDK for Java 2.x を使用した Amazon Cognito ID プロバイダーの例

次のコード例は、Amazon Cognito ID プロバイダー AWS SDK for Java 2.x でを使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

## 開始方法

### Hello Amazon Cognito

次のコード例は、Amazon Cognito の使用を開始する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
            CognitoIdentityProviderClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    listAllUserPools(cognitoClient);
    cognitoClient.close();
}

public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient)
{
    try {
        ListUserPoolsRequest request = ListUserPoolsRequest.builder()
            .maxResults(10)
            .build();

        ListUserPoolsResponse response = cognitoClient.listUserPools(request);
        response.userPools().forEach(userpool -> {
            System.out.println("User pool " + userpool.name() + ", User ID " +
                userpool.id());
        });

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、「APIリファレンス[ListUserPools](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### AdminGetUser

次の例は、AdminGetUser を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
    String poolId) {
    try {
        AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
            .username(userName)
            .userPoolId(poolId)
            .build();

        AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
        System.out.println("User status " + response.userStatusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [AdminGetUser](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## AdminInitiateAuth

次の例は、AdminInitiateAuth を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
    String clientId, String userName, String password, String userPoolId) {
    try {
        Map<String, String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
    .clientId(clientId)
    .userPoolId(userPoolId)
    .authParameters(authParameters)
    .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
    .build();

        AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
        System.out.println("Result Challenge is : " + response.challengeName());
        return response;

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- API の詳細については、「API リファレンス [AdminInitiateAuth](#)」の「」を参照してください。  
AWS SDK for Java 2.x



## AdminRespondToAuthChallenge

次の例は、AdminRespondToAuthChallenge を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Respond to an authentication challenge.
public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
    String userName, String clientId, String mfaCode, String session) {
    System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
    Map<String, String> challengeResponses = new HashMap<>();

    challengeResponses.put("USERNAME", userName);
    challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

    AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
        .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
        .clientId(clientId)
        .challengeResponses(challengeResponses)
        .session(session)
        .build();

    AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
        .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
    System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
        + respondToAuthChallengeResult.authenticationResult());
}
```

- API の詳細については、「API リファレンス [AdminRespondToAuthChallenge](#)」の「」を参照してください。 AWS SDK for Java 2.x

## AssociateSoftwareToken

次の例は、AssociateSoftwareToken を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
        .session(session)
        .build();

    AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
        .associateSoftwareToken(softwareTokenRequest);
    String secretCode = tokenResponse.secretCode();
    System.out.println("Enter this token into Google Authenticator");
    System.out.println(secretCode);
    return tokenResponse.session();
}
```

- API の詳細については、「API リファレンス [AssociateSoftwareToken](#)」の「」を参照してください。 AWS SDK for Java 2.x

## ConfirmSignUp

次の例は、ConfirmSignUp を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
    String userName) {
    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
            .username(userName)
            .build();

        identityProviderClient.confirmSignUp(signUpRequest);
        System.out.println(userName + " was confirmed");

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [ConfirmSignUp](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateUserPool

次の例は、CreateUserPool を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUserPool {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <userPoolName>\s

            Where:
                userPoolName - The name to give your user pool when it's
created.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String userPoolName = args[0];
    CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
        .region(Region.US_EAST_1)
        .build();

    String id = createPool(cognitoClient, userPoolName);
    System.out.println("User pool ID: " + id);
    cognitoClient.close();
}

public static String createPool(CognitoIdentityProviderClient cognitoClient,
String userPoolName) {
    try {
        CreateUserPoolRequest request = CreateUserPoolRequest.builder()
            .poolName(userPoolName)
            .build();

        CreateUserPoolResponse response = cognitoClient.createUserPool(request);
        return response.userPool().id();

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- API の詳細については、「API リファレンス [CreateUserPool](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateUserPoolClient

次の例は、CreateUserPoolClient を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientResponse;

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <clientName> <userPoolId>\s

            Where:
                clientName - The name for the user pool client to create.
```

```
        userPoolId - The ID for the user pool.
        """);

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String clientName = args[0];
    String userPoolId = args[1];
    CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
        .region(Region.US_EAST_1)
        .build();

    createPoolClient(cognitoClient, clientName, userPoolId);
    cognitoClient.close();
}

public static void createPoolClient(CognitoIdentityProviderClient cognitoClient,
String clientName,
    String userPoolId) {
    try {
        CreateUserPoolClientRequest request =
CreateUserPoolClientRequest.builder()
            .clientName(clientName)
            .userPoolId(userPoolId)
            .build();

        CreateUserPoolClientResponse response =
cognitoClient.createUserPoolClient(request);
        System.out.println("User pool " + response.userPoolClient().clientName()
+ " created. ID: "
            + response.userPoolClient().clientId());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [CreateUserPoolClient](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListUserPools

次の例は、ListUserPools を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
        CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }
}
```



```
    }

    public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient)
    {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                .maxResults(10)
                .build();

            ListUserPoolsResponse response = cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID " +
                    userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [ListUserPools](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListUsers

次の例は、ListUsers を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import
software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
```

```
import
software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersRequest;
import
software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUsers {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <userPoolId>\s

            Where:
                userPoolId - The ID given to your user pool when it's created.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolId = args[0];
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
        cognitoClient.close();
    }
}
```

```
public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {
    try {
        ListUsersRequest usersRequest = ListUsersRequest.builder()
            .userPoolId(userPoolId)
            .build();

        ListUsersResponse response = cognitoClient.listUsers(usersRequest);
        response.users().forEach(user -> {
            System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created "
                + user.userCreateDate());
        });

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Shows how to list users by using a filter.
public static void listUsersFilter(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {

    try {
        String filter = "email = \"tblue@noserver.com\"";
        ListUsersRequest usersRequest = ListUsersRequest.builder()
            .userPoolId(userPoolId)
            .filter(filter)
            .build();

        ListUsersResponse response = cognitoClient.listUsers(usersRequest);
        response.users().forEach(user -> {
            System.out.println("User with filter applied " + user.username() + "
Status " + user.userStatus()
                + " Created " + user.userCreateDate());
        });

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [ListUsers](#)」の「」を参照してください。AWS SDK for Java 2.x

## ResendConfirmationCode

次の例は、ResendConfirmationCode を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
    String userName) {
    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
            .clientId(clientId)
            .username(userName)
            .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [ResendConfirmationCode](#)」の「」を参照してください。AWS SDK for Java 2.x

## SignUp

次の例は、SignUp を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
    String password, String email) {
    AttributeType userAttrs = AttributeType.builder()
        .name("email")
        .value(email)
        .build();

    List<AttributeType> userAttrsList = new ArrayList<>();
    userAttrsList.add(userAttrs);
    try {
        SignUpRequest signUpRequest = SignUpRequest.builder()
            .userAttributes(userAttrsList)
            .username(userName)
            .clientId(clientId)
            .password(password)
            .build();

        identityProviderClient.signUp(signUpRequest);
        System.out.println("User has been signed up ");

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [SignUp](#)」の「」を参照してください。AWS SDK for Java 2.x

## VerifySoftwareToken

次の例は、VerifySoftwareToken を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
            .userCode(code)
            .session(session)
            .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is " +
verifyResponse.statusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [VerifySoftwareToken](#)」の「」を参照してください。 AWS SDK for Java 2.x

## シナリオ

MFA を必要とするユーザープールによりユーザーをサインアップする

次のコードサンプルは、以下の操作方法を示しています。

- ユーザー名、パスワード、E メールアドレスでサインアップしてユーザーを確認します。
- MFA アプリケーションをユーザーに関連付けて、多要素認証を設定します。
- パスワードと MFA コードを使用してサインインします。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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.AdminRespondToAuthChallengeRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenResponse;
import software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
```

```
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeResponse;
import software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenResponse;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito\_scenario\_user\_pool\_with\_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 * to set up TOTP (time-based one-time password). (The response is
 * "ChallengeName": "MFA_SETUP").
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
```



```

* key. This can be used with Google Authenticator.
* 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
* MFA.
* 8. Invokes the AdminInitiateAuth to sign in again. This results in being
* prompted to submit a TOTP (Response: "ChallengeName": "SOFTWARE_TOKEN_MFA").
* 9. Invokes the AdminRespondToAuthChallenge to get back a token.
*/

```

```

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws NoSuchAlgorithmException,
    InvalidKeyException {
        final String usage = ""

            Usage:
                <clientId> <poolId>

            Where:
                clientId - The app client Id value that you can get from the AWS
CDK script.
                poolId - The pool Id that you can get from the AWS CDK script.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

```

```
System.out.println("*** Enter your password");
String password = in.nextLine();

System.out.println("*** Enter your email");
String email = in.nextLine();

System.out.println("1. Signing up " + userName);
signUp(identityProviderClient, clientId, userName, password, email);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Getting " + userName + " in the user pool");
getAdminUser(identityProviderClient, userName, poolId);

System.out
    .println("*** Conformation code sent to " + userName + ". Would you
like to send a new code? (Yes/No)");
System.out.println(DASHES);

System.out.println(DASHES);
String ans = in.nextLine();

if (ans.compareTo("Yes") == 0) {
    resendConfirmationCode(identityProviderClient, clientId, userName);
    System.out.println("3. Sending a new confirmation code");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Enter confirmation code that was emailed");
String code = in.nextLine();
confirmSignUp(identityProviderClient, clientId, code, userName);
System.out.println("Rechecking the status of " + userName + " in the user
pool");
getAdminUser(identityProviderClient, userName, poolId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Invokes the initiateAuth to sign in");
AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
    poolId);
String mySession = authResponse.session();
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to generate
a TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient, mySession);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
        AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
```

```
        String userName, String clientId, String mfaCode, String session) {
    System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
    Map<String, String> challengeResponses = new HashMap<>();

    challengeResponses.put("USERNAME", userName);
    challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

    AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
        .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
        .clientId(clientId)
        .challengeResponses(challengeResponses)
        .session(session)
        .build();

    AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
        .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
    System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
        + respondToAuthChallengeResult.authenticationResult());
}

// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
            .userCode(code)
            .session(session)
            .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is " +
verifyResponse.statusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
             String clientId, String userName, String password, String userPoolId) {
    try {
        Map<String, String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                        .clientId(clientId)
                        .userPoolId(userPoolId)
                        .authParameters(authParameters)
                        .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                        .build();

        AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
        System.out.println("Result Challenge is :" + response.challengeName());
        return response;

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}

public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
                        .session(session)
                        .build();

    AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
        .associateSoftwareToken(softwareTokenRequest);
    String secretCode = tokenResponse.secretCode();
    System.out.println("Enter this token into Google Authenticator");
    System.out.println(secretCode);
    return tokenResponse.session();
}
```

```
public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
    String userName) {
    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
            .username(userName)
            .build();

        identityProviderClient.confirmSignUp(signUpRequest);
        System.out.println(userName + " was confirmed");

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
    String userName) {
    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
            .clientId(clientId)
            .username(userName)
            .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
    String password, String email) {
    AttributeType userAttrs = AttributeType.builder()
```

```
        .name("email")
        .value(email)
        .build();

List<AttributeType> userAttrsList = new ArrayList<>();
userAttrsList.add(userAttrs);
try {
    SignUpRequest signUpRequest = SignUpRequest.builder()
        .userAttributes(userAttrsList)
        .username(userName)
        .clientId(clientId)
        .password(password)
        .build();

    identityProviderClient.signUp(signUpRequest);
    System.out.println("User has been signed up ");

} catch (CognitoIdentityProviderException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
String poolId) {
    try {
        AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
            .username(userName)
            .userPoolId(poolId)
            .build();

        AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
        System.out.println("User status " + response.userStatusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [AdminGetUser](#)
  - [AdminInitiateAuth](#)
  - [AdminRespondToAuthChallenge](#)
  - [AssociateSoftwareToken](#)
  - [ConfirmDevice](#)
  - [ConfirmSignUp](#)
  - [InitiateAuth](#)
  - [ListUsers](#)
  - [ResendConfirmationCode](#)
  - [RespondToAuthChallenge](#)
  - [SignUp](#)
  - [VerifySoftwareToken](#)

## SDK for Java 2.x を使用する Amazon Comprehend の例

次のコード例は、Amazon Comprehend AWS SDK for Java 2.x を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

### トピック

- [アクション](#)



## アクション

### CreateDocumentClassifier

次の例は、CreateDocumentClassifier を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「API リファレンス [CreateDocumentClassifier](#)」の「」を参照してください。AWS SDK for Java 2.x

## DetectDominantLanguage

次の例は、DetectDominantLanguage を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「API リファレンス [DetectDominantLanguage](#)」の「」を参照してください。AWS SDK for Java 2.x

## DetectEntities

次の例は、DetectEntities を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「API リファレンス [DetectEntities](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DetectKeyPhrases

次の例は、DetectKeyPhrases を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
}
}
}
```

- API の詳細については、「API リファレンス [DetectKeyPhrases](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DetectSentiment

次の例は、DetectSentiment を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- APIの詳細については、「APIリファレンス[DetectSentiment](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DetectSyntax

次の例は、DetectSyntax を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [DetectSyntax](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## SDK for Java 2.x を使用した DynamoDB の例

次のコード例は、DynamoDB AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

## 開始方法

### Hello DynamoDB

次のコード例は、DynamoDB の使用を開始する方法を示しています。

#### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
```

```
String lastName = null;

while (moreTables) {
    try {
        ListTablesResponse response = null;
        if (lastName == null) {
            ListTablesRequest request = ListTablesRequest.builder().build();
            response = ddb.listTables(request);
        } else {
            ListTablesRequest request = ListTablesRequest.builder()
                .exclusiveStartTableName(lastName).build();
            response = ddb.listTables(request);
        }

        List<String> tableNames = response.tableNames();
        if (tableNames.size() > 0) {
            for (String curName : tableNames) {
                System.out.format("* %s\n", curName);
            }
        } else {
            System.out.println("No tables found!");
            System.exit(0);
        }

        lastName = response.lastEvaluatedTableName();
        if (lastName == null) {
            moreTables = false;
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
System.out.println("\nDone!");
}
```

- API の詳細については、「API リファレンス [ListTables](#)」の「」を参照してください。AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)
- [サーバーレスサンプル](#)

## アクション

### BatchGetItem

次の例は、BatchGetItem を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

サービスクライアントを使用してバッチアイテムを取得する方法を示します。

```
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.");
    }
}
}
```

サービスクライアントおよびページネーターを使用してバッチアイテムを取得する方法を示します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class BatchGetItemsPaginator {

    public static void main(String[] args){
        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table (for example, Music).\s
            """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();
```



```
        getBatchItemsPaginator(dynamoDbClient, tableName) ;
    }

    public static void getBatchItemsPaginator(DynamoDbClient dynamoDbClient, String
tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
        key1.put("Artist", AttributeValue.builder().s("Artist1").build());

        Map<String, AttributeValue> key2 = new HashMap<>();
        key2.put("Artist", AttributeValue.builder().s("Artist2").build());

        // Construct the batchGetItem request.
        Map<String, KeysAndAttributes> requestItems = new HashMap<>();
        requestItems.put(tableName, KeysAndAttributes.builder()
            .keys(List.of(key1, key2))
            .projectionExpression("Artist, SongTitle")
            .build());

        BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
            .requestItems(requestItems)
            .build();

        // Use batchGetItemPaginator for paginated requests.
        dynamoDbClient.batchGetItemPaginator(batchGetItemRequest).stream()
            .flatMap(response -> response.responses().getOrDefault(tableName,
Collections.emptyList()).stream())
            .forEach(item -> {
                System.out.println("Artist: " + item.get("Artist").s() +
                    ", SongTitle: " + item.get("SongTitle").s());
            });
    }
}
```

- API の詳細については、「API リファレンス [BatchGetItem](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## BatchWriteItem

次の例は、BatchWriteItem を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

サービスクライアントを使用して、多数の項目をテーブルに挿入します。

```
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);
    }
}
}
```

拡張クライアントを使用して、多数の項目をテーブルに挿入します。

```
import com.example.dynamodb.Customer;
import com.example.dynamodb.Music;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.Key;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.enhanced.dynamodb.model.BatchWriteItemEnhancedRequest;
import software.amazon.awssdk.enhanced.dynamodb.model.WriteBatch;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/*
 * Before running this code example, create an Amazon DynamoDB table named Customer
 * with these columns:
 * - id - the id of the record that is the key
 * - custName - the customer name
 * - email - the email value
 * - registrationDate - an instant value when the item was added to the table
 *
 * Also, ensure that you have set up your development environment, including your
 * credentials.
```

```
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class EnhancedBatchWriteItems {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
        DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();
        putBatchRecords(enhancedClient);
        ddb.close();
    }

    public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient) {
        try {
            DynamoDbTable<Customer> customerMappedTable =
enhancedClient.table("Customer",
                TableSchema.fromBean(Customer.class));
            DynamoDbTable<Music> musicMappedTable =
enhancedClient.table("Music",
                TableSchema.fromBean(Music.class));
            LocalDate localDate = LocalDate.parse("2020-04-07");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            Customer record2 = new Customer();
            record2.setCustName("Fred Pink");
            record2.setId("id110");
            record2.setEmail("fredp@noserver.com");
            record2.setRegistrationDate(instant);

            Customer record3 = new Customer();
            record3.setCustName("Susan Pink");
            record3.setId("id120");
            record3.setEmail("spink@noserver.com");
            record3.setRegistrationDate(instant);

            Customer record4 = new Customer();
```

```

        record4.setCustName("Jerry orange");
        record4.setId("id101");
        record4.setEmail("jorange@noserver.com");
        record4.setRegistrationDate(instant);

        BatchWriteItemEnhancedRequest batchWriteItemEnhancedRequest
= BatchWriteItemEnhancedRequest
                                .builder()
                                .writeBatches(

WriteBatch.builder(Customer.class) // add items to the Customer

        // table

        .mappedTableResource(customerMappedTable)

        .addPutItem(builder -> builder.item(record2))

        .addPutItem(builder -> builder.item(record3))

        .addPutItem(builder -> builder.item(record4))

                                                                .build(),

WriteBatch.builder(Music.class) // delete an item from the Music

        // table

        .mappedTableResource(musicMappedTable)

        .addDeleteItem(builder -> builder.key(

            Key.builder().partitionValue(

                "Famous Band")

                .build()))

                                                                .build())

                                                                .build();

        // Add three items to the Customer table and delete one item
from the Music

        // table.

enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);

```

```
        System.out.println("done");
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [BatchWriteItem](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateTable

次の例は、CreateTable を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateTable {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <key>

            Where:
                tableName - The Amazon DynamoDB table to create (for example,
Music3).
                key - The key for the Amazon DynamoDB table (for example,
Artist).

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        System.out.println("Creating an Amazon DynamoDB table " + tableName + " with
a simple primary key: " + key);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        String result = createTable(ddb, tableName, key);
        System.out.println("New table is " + result);
        ddb.close();
    }

    public static String createTable(DynamoDbClient ddb, String tableName, String
key) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        CreateTableRequest request = CreateTableRequest.builder()
```



```
        .attributeDefinitions(AttributeDefinition.builder()
            .attributeName(key)
            .attributeType(ScalarAttributeType.S)
            .build())
        .keySchema(KeySchemaElement.builder()
            .attributeName(key)
            .keyType(KeyType.HASH)
            .build())
        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(10L)
            .writeCapacityUnits(10L)
            .build())
        .tableName(tableName)
        .build();

String newTable;
try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    newTable = response.tableDescription().tableName();
    return newTable;

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
}
```

- APIの詳細については、「APIリファレンス[CreateTable](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteItem

次の例は、DeleteItem を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteItem {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <key> <keyval>

            Where:
                tableName - The Amazon DynamoDB table to delete the item from
                (for example, Music3).
                key - The key used in the Amazon DynamoDB table (for example,
                Artist).\s
                keyval - The key value that represents the item to delete (for
                example, Famous Band).
        """;
```

```
        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Deleting item \"%s\" from %s\n", keyVal, tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        deleteDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }

    public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName,
String key, String keyVal) {
        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put(key, AttributeValue.builder()
            .s(keyVal)
            .build());

        DeleteItemRequest deleteReq = DeleteItemRequest.builder()
            .tableName(tableName)
            .key(keyToGet)
            .build();

        try {
            ddb.deleteItem(deleteReq);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [DeleteItem](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteTable

次の例は、DeleteTable を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteTable {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table to delete (for example,
Music3).

            **Warning** This program will delete the table that you specify!
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String tableName = args[0];
    System.out.format("Deleting the Amazon DynamoDB table %s...\n", tableName);
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    deleteDynamoDBTable(ddb, tableName);
    ddb.close();
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}
}
```

- APIの詳細については、「APIリファレンス[DeleteTable](#)」の「」を参照してください。AWS SDK for Java 2.x

## DescribeTable

次の例は、DescribeTable を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import
    software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeTable {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <tableName>

                Where:
                tableName - The Amazon DynamoDB table to get information about
                (for example, Music3).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String tableName = args[0];
System.out.format("Getting description for %s\n\n", tableName);
Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();

describeDynamoDBTable(ddb, tableName);
ddb.close();
}

public static void describeDynamoDBTable(DynamoDbClient ddb, String tableName) {
    DescribeTableRequest request = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

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

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

            List<AttributeDefinition> attributes =
tableInfo.attributeDefinitions();
            System.out.println("Attributes");
            for (AttributeDefinition a : attributes) {
                System.out.format("  %s (%s)\n", a.attributeName(),
a.attributeType());
            }
        }
    } catch (DynamoDbException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("\nDone!");
}
}
```

- API の詳細については、「API リファレンス [DescribeTable](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DescribeTimeToLive

次の例は、DescribeTimeToLive を使用する方法を説明しています。

SDK for Java 2.x

既存の DynamoDB テーブルの TTL 設定について説明します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.Optional;

    final DescribeTimeToLiveRequest request =
DescribeTimeToLiveRequest.builder()
        .tableName(tableName)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final DescribeTimeToLiveResponse response =
ddb.describeTimeToLive(request);
        System.out.println(tableName + " description of time to live is "
            + response.toString());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    }
```



```
    } catch (DynamoDbException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
    System.exit(0);
```

- API の詳細については、「API リファレンス [DescribeTimeToLive](#)」の「」を参照してください。AWS SDK for Java 2.x

## GetItem

次の例は、GetItem を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

を使用してテーブルから項目を取得します 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);
}
}
```

- APIの詳細については、「APIリファレンス[GetItem](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListTables

次の例は、ListTables を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request = ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                        .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();
            }
        }
    }
}
```

```
        if (tableNames.size() > 0) {
            for (String curName : tableNames) {
                System.out.format("* %s\n", curName);
            }
        } else {
            System.out.println("No tables found!");
            System.exit(0);
        }

        lastName = response.lastEvaluatedTableName();
        if (lastName == null) {
            moreTables = false;
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
System.out.println("\nDone!");
}
```

- API の詳細については、「API リファレンス [ListTables](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutItem

次の例は、PutItem を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

を使用してテーブルに項目を配置します [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);
        }
    }
}
```

- APIの詳細については、「APIリファレンス[PutItem](#)」の「」を参照してください。AWS SDK for Java 2.x

## Query

次の例は、Query を使用する方法を説明しています。



## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

を使用してテーブルをクエリします [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;
}
}
```

DynamoDbClient とセカンダリインデックスを使用してテーブルに対してクエリを実行します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Create the Movies table by running the Scenario example and loading the Movie
 * data from the JSON file. Next create a secondary
 * index for the Movies table that uses only the year column. Name the index
```

```
* **year-index**. For more information, see:
*
* https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html
*/
public class QueryItemsUsingIndex {
    public static void main(String[] args) {
        String tableName = "Movies";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        queryIndex(ddb, tableName);
        ddb.close();
    }

    public static void queryIndex(DynamoDbClient ddb, String tableName) {
        try {
            Map<String, String> expressionAttributesNames = new HashMap<>();
            expressionAttributesNames.put("#year", "year");
            Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
            expressionAttributeValues.put(":yearValue",
AttributeValue.builder().n("2013").build());

            QueryRequest request = QueryRequest.builder()
                .tableName(tableName)
                .indexName("year-index")
                .keyConditionExpression("#year = :yearValue")
                .expressionAttributeNames(expressionAttributesNames)
                .expressionAttributeValues(expressionAttributeValues)
                .build();

            System.out.println("=== Movie Titles ===");
            QueryResponse response = ddb.query(request);
            response.items()
                .forEach(movie -> System.out.println(movie.get("title").s()));

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[Query](#)」を参照してください。

## Scan

次の例は、Scan を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

を使用して Amazon DynamoDB テーブルをスキャンします [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);
    }
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[Scan](#)」を参照してください。

## UpdateItem

次の例は、UpdateItem を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

を使用してテーブル内の項目を更新します [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!");
}
}
```

- API の詳細については、「API リファレンス [UpdateItem](#)」の「」を参照してください。AWS SDK for Java 2.x

## UpdateTimeToLive

次の例は、UpdateTimeToLive を使用する方法を説明しています。

SDK for Java 2.x

既存の DynamoDB テーブルで TTL を有効にします。

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

```
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.Optional;

    final TimeToLiveSpecification ttlSpecification =
TimeToLiveSpecification.builder()
    .attributeName(ttlAttributeName)
    .enabled(true)
    .build();
    final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
    .tableName(tableName)
    .timeToLiveSpecification(ttlSpecification)
    .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build()) {
        final UpdateTimeToLiveResponse response =
ddb.updateTimeToLive(request);
        System.out.println(tableName + " had its TTL successfully updated.
The request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done!");
```

既存の DynamoDB テーブルで TTL を無効にします。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
```

```
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.Optional;

    final Region region = Optional.ofNullable(args[2]).isEmpty() ?
Region.US_EAST_1 : Region.of(args[2]);
    final TimeToLiveSpecification ttlSpecification =
TimeToLiveSpecification.builder()
        .attributeName(ttlAttributeName)
        .enabled(false)
        .build();
    final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
        .tableName(tableName)
        .timeToLiveSpecification(ttlSpecification)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final UpdateTimeToLiveResponse response = ddb.updateTimeToLive(request);
        System.out.println(tableName + " had its TTL successfully updated. The
request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done!");
```

- API の詳細については、「API リファレンス [UpdateTimeToLive](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## シナリオ

### 項目の TTL を条件付きで更新する

次のコード例は、項目の TTL を条件付きで更新する方法を示しています。

## SDK for Java 2.x

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.amazon.samplelib.ttl;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
import java.util.Optional;

public class UpdateTTLConditional {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <tableName> <primaryKey> <sortKey> <newTtlAttribute> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash
or partition key.
                sortKey - The name of the sort key. Also known as the range
attribute.
                newTtlAttribute - New attribute name (as part of the update
command)
                region (optional) - The AWS region that the Amazon DynamoDB
table is located in. (Default: us-east-1)
            """;
        // Optional "region" parameter - if args list length is NOT 3 or 4, short-
circuit exit.
        if (!(args.length == 4 || args.length == 5)) {
            System.out.println(usage);
            System.exit(1);
        }
        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final String newTtlAttribute = args[3];
```

```

    Region region = Optional.ofNullable(args[4]).isEmpty() ? Region.US_EAST_1 :
    Region.of(args[4]);

    // Get current time in epoch second format
    final long currentTime = System.currentTimeMillis() / 1000;
    // Calculate expiration time 90 days from now in epoch second format
    final long expireDate = currentTime + (90 * 24 * 60 * 60);
    // An expression that defines one or more attributes to be updated, the
    action to be performed on them, and new values for them.
    final String updateExpression = "SET newTtlAttribute = :val1";
    // A condition that must be satisfied in order for a conditional update to
    succeed.
    final String conditionExpression = "expireAt > :val2";

    final ImmutableMap<String, AttributeValue> keyMap =
        ImmutableMap.of("primaryKey", AttributeValue.fromS(primaryKey),
            "sortKey", AttributeValue.fromS(sortKey));
    final Map<String, AttributeValue> expressionAttributeValues =
    ImmutableMap.of(
        ":val1", AttributeValue.builder().s(newTtlAttribute).build(),
        ":val2",
    AttributeValue.builder().s(String.valueOf(expireDate)).build()
    );

    final UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(keyMap)
        .updateExpression(updateExpression)
        .conditionExpression(conditionExpression)
        .expressionAttributeValues(expressionAttributeValues)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final UpdateItemResponse response = ddb.updateItem(request);
        System.out.println(tableName + " UpdateItem operation with conditional
    TTL successful. Request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
    found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
    }

```

```
        System.exit(1);
    }
    System.exit(0);
}
}
```

- API の詳細については、「API リファレンス [UpdateItem](#)」の「」を参照してください。AWS SDK for Java 2.x

## TTL を使用して項目を作成する

次のコード例は、TTL を使用して項目を作成する方法を示しています。

### SDK for Java 2.x

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

package com.amazon.samplelib.ttl;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.utils.ImmutableMap;

import java.io.Serializable;
import java.util.Map;
import java.util.Optional;

public class CreateTTL {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <tableName> <primaryKey> <sortKey> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash
or partition key.
```

```

        sortKey - The name of the sort key. Also known as the range
attribute.
        region (optional) - The AWS region that the Amazon DynamoDB
table is located in. (Default: us-east-1)
        """;
    // Optional "region" parameter - if args list length is NOT 3 or 4, short-
circuit exit.
    if (!(args.length == 3 || args.length == 4)) {
        System.out.println(usage);
        System.exit(1);
    }

    String tableName = args[0];
    String primaryKey = args[1];
    String sortKey = args[2];
    Region region = Optional.ofNullable(args[3]).isEmpty() ? Region.US_EAST_1 :
Region.of(args[3]);

    // Get current time in epoch second format
    final long createDate = System.currentTimeMillis() / 1000;

    // Calculate expiration time 90 days from now in epoch second format
    final long expireDate = createDate + (90 * 24 * 60 * 60);

    final ImmutableMap<String, ? extends Serializable> itemMap =
        ImmutableMap.of("primaryKey", primaryKey,
            "sortKey", sortKey,
            "creationDate", createDate,
            "expireAt", expireDate);
    final PutItemRequest request = PutItemRequest.builder()
        .tableName(tableName)
        .item((Map<String, AttributeValue>) itemMap)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final PutItemResponse response = ddb.putItem(request);
        System.out.println(tableName + " PutItem operation with TTL successful.
Request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    }

```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.exit(0);
    }
}
```


- API の詳細については、「API リファレンス [PutItem](#)」の「」を参照してください。AWS SDK for Java 2.x

テーブル、項目、クエリで使用を開始する

次のコードサンプルは、以下の操作方法を示しています。

- 映画データを保持できるテーブルを作成する。
- テーブルに 1 つの映画を入れ、取得して更新する。
- サンプル JSON ファイルから映画データをテーブルに書き込む。
- 特定の年にリリースされた映画を照会する。
- 何年もの間にリリースされた映画をスキャンする。
- テーブルからムービーを削除し、テーブルを削除します。

SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

DynamoDB テーブルを作成します。

```
// Create a table with a Sort key.
public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
```



```
attributeDefinitions.add(AttributeDefinition.builder()
    .attributeName("year")
    .attributeType("N")
    .build());

attributeDefinitions.add(AttributeDefinition.builder()
    .attributeName("title")
    .attributeType("S")
    .build());

ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
KeySchemaElement key = KeySchemaElement.builder()
    .attributeName("year")
    .keyType(KeyType.HASH)
    .build();

KeySchemaElement key2 = KeySchemaElement.builder()
    .attributeName("title")
    .keyType(KeyType.RANGE)
    .build();

// Add KeySchemaElement objects to the list.
tableKey.add(key);
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(10L)
        .writeCapacityUnits(10L)
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
    dbWaiter.waitUntilTableExists(tableRequest);
```

```
        waiterResponse.matched().response().ifPresent(System.out::println);
        String newTable = response.tableDescription().tableName();
        System.out.println("The " + newTable + " was successfully created.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

サンプルの JSON ファイルをダウンロードして抽出するヘルパー関数を作成します。

```
// 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++;
    }
}
```

テーブルから項目を取得します。

```
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);
    }
}
```

```
    }  
}
```

完全な例です。

```
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * This Java example performs these tasks:  
 *  
 * 1. Creates the Amazon DynamoDB Movie table with partition and sort key.  
 * 2. Puts data into the Amazon DynamoDB table from a JSON document using the  
 * Enhanced client.  
 * 3. Gets data from the Movie table.  
 * 4. Adds a new item.  
 * 5. Updates an item.  
 * 6. Uses a Scan to query items using the Enhanced client.  
 * 7. Queries all items where the year is 2013 using the Enhanced Client.  
 * 8. Deletes the table.  
 */  
  
public class Scenario {  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
  
    public static void main(String[] args) throws IOException {  
        final String usage = ""  
  
            Usage:  
            <fileName>  
  
        Where:  
            fileName - The path to the moviedata.json file that you can  
download from the Amazon DynamoDB Developer Guide.  
            "";  
  
        if (args.length != 1) {  
            System.out.println(usage);  
        }  
    }  
}
```

```
        System.exit(1);
    }

    String tableName = "Movies";
    String fileName = args[0];
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon DynamoDB example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println(
        "1. Creating an Amazon DynamoDB table named Movies with a key named
year and a sort key named title.");
    createTable(ddb, tableName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Loading data into the Amazon DynamoDB table.");
    loadData(ddb, tableName, fileName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Getting data from the Movie table.");
    getItem(ddb);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Putting a record into the Amazon DynamoDB table.");
    putRecord(ddb);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("5. Updating a record.");
    updateTableItem(ddb, tableName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("6. Scanning the Amazon DynamoDB table.");
    scanMovies(ddb, tableName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Querying the Movies released in 2013.");
        queryTable(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        System.out.println(DASHES);

        ddb.close();
    }

    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE)
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
```

```
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(10L)
        .writeCapacityUnits(10L)
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " + newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Query the table.
public static void queryTable(DynamoDbClient ddb) {
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        QueryConditional queryConditional = QueryConditional
            .keyEqualTo(Key.builder()
                .partitionValue(2013)
                .build());
```

```
        // Get items in the table and write out the ID value.
        Iterator<Movies> results =
custTable.query(queryConditional).items().iterator();
        String result = "";

        while (results.hasNext()) {
            Movies rec = results.next();
            System.out.println("The title of the movie is " + rec.getTitle());
            System.out.println("The movie information is " + rec.getInfo());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Scan the table.
public static void scanMovies(DynamoDbClient ddb, String tableName) {
    System.out.println("***** Scanning all movies.\n");
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        Iterator<Movies> results = custTable.scan().items().iterator();
        while (results.hasNext()) {
            Movies rec = results.next();
            System.out.println("The movie title is " + rec.getTitle());
            System.out.println("The movie year is " + rec.getYear());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
```



```

        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {
            // Only add 200 Movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            Movies movies = new Movies();
            movies.setYear(year);
            movies.setTitle(title);
            movies.setInfo(info);

            // Put the data into the Amazon DynamoDB Movie table.
            mappedTable.putItem(movies);
            t++;
        }
    }

    // Update the record to include show only directors.
    public static void updateTableItem(DynamoDbClient ddb, String tableName) {
        HashMap<String, AttributeValue> itemKey = new HashMap<>();
        itemKey.put("year", AttributeValue.builder().n("1933").build());
        itemKey.put("title", AttributeValue.builder().s("King Kong").build());

        HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
        updatedValues.put("info", AttributeValueUpdate.builder()
            .value(AttributeValue.builder().s("{\"directors\": [\"Merian C.
Cooper\", \"Ernest B. Schoedsack\"]")
            .build())

```

```
        .action(AttributeAction.PUT)
        .build());

UpdateItemRequest request = UpdateItemRequest.builder()
    .tableName(tableName)
    .key(itemKey)
    .attributeUpdates(updatedValues)
    .build();

try {
    ddb.updateItem(request);
} catch (ResourceNotFoundException e) {
    System.err.println(e.getMessage());
    System.exit(1);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

System.out.println("Item was updated!");
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}

public static void putRecord(DynamoDbClient ddb) {
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();
    }
}
```

```
DynamoDbTable<Movies> table = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));

// Populate the Table.
Movies record = new Movies();
record.setYear(2020);
record.setTitle("My Movie2");
record.setInfo("no info");
table.putItem(record);

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("Added a new movie to the table.");
}

public static void getItem(DynamoDbClient ddb) {

    HashMap<String, AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put("year", AttributeValue.builder()
        .n("1933")
        .build());

    keyToGet.put("title", AttributeValue.builder()
        .s("King Kong")
        .build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName("Movies")
        .build();

    try {
        Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();

        if (returnedItem != null) {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");

            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
            }
        }
    }
}
```

```
        } else {
            System.out.format("No item found with the key %s!\n", "year");
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

PartiQL ステートメントのバッチを使用してテーブルにクエリを実行する

次のコードサンプルは、以下の操作方法を示しています。

- 複数の SELECT ステートメントを実行して、項目のバッチを取得する。
- 複数の INSERT ステートメントを実行して、項目のバッチを追加する。
- 複数の UPDATE ステートメントを実行して、項目のバッチを更新する。
- 複数の DELETE ステートメントを実行して、項目のバッチを削除する。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public class ScenarioPartiQLBatch {
    public static void main(String[] args) throws IOException {
        String tableName = "MoviesPartiQBatch";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println("***** Creating an Amazon DynamoDB table named
" + tableName
            + " with a key named year and a sort key named
title.");
        createTable(ddb, tableName);

        System.out.println("***** Adding multiple records into the " +
tableName
            + " table using a batch command.");
        putRecordBatch(ddb);

        System.out.println("***** Updating multiple records using a batch
command.");
        updateTableItemBatch(ddb);

        System.out.println("***** Deleting multiple records using a batch
command.");
        deleteItemBatch(ddb);

        System.out.println("***** Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
```

```
        ArrayList<AttributeDefinition> attributeDefinitions = new
ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)

        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10))
            .build())
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest =
DescribeTableRequest.builder()
```

```
                .tableName(tableName)
                .build();

        // Wait until the Amazon DynamoDB table is created.
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter
                .waitUntilTableExists(tableRequest);

waiterResponse.matched().response().ifPresent(System.out::println);
        String newTable = response.tableDescription().tableName();
        System.out.println("The " + newTable + " was successfully
created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecordBatch(DynamoDbClient ddb) {
        String sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE
{'year':?, 'title' : ?, 'info' : ?}";
        try {
            // Create three movies to add to the Amazon DynamoDB table.
            // Set data for Movie 1.
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2022"))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("My Movie 1")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);
```

```
BatchStatementRequest statementRequestMovie1 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parameters)
    .build();

// Set data for Movie 2.
List<AttributeValue> parametersMovie2 = new ArrayList<>();
AttributeValue attMovie2 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attMovie2A = AttributeValue.builder()
    .s("My Movie 2")
    .build();

AttributeValue attMovie2B = AttributeValue.builder()
    .s("No Information")
    .build();

parametersMovie2.add(attMovie2);
parametersMovie2.add(attMovie2A);
parametersMovie2.add(attMovie2B);

BatchStatementRequest statementRequestMovie2 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersMovie2)
    .build();

// Set data for Movie 3.
List<AttributeValue> parametersMovie3 = new ArrayList<>();
AttributeValue attMovie3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attMovie3A = AttributeValue.builder()
    .s("My Movie 3")
    .build();

AttributeValue attMovie3B = AttributeValue.builder()
    .s("No Information")
    .build();
```



```

        parametersMovie3.add(attMovie3);
        parametersMovie3.add(attMovie3A);
        parametersMovie3.add(attMovie3B);

        BatchStatementRequest statementRequestMovie3 =
BatchStatementRequest.builder()
                        .statement(sqlStatement)
                        .parameters(parametersMovie3)
                        .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();

        myBatchStatementList.add(statementRequestMovie1);
        myBatchStatementList.add(statementRequestMovie2);
        myBatchStatementList.add(statementRequestMovie3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
                        .statements(myBatchStatementList)
                        .build();

        BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
        System.out.println("ExecuteStatement successful: " +
response.toString());
        System.out.println("Added new movies using a batch
command.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateTableItemBatch(DynamoDbClient ddb) {
    String sqlStatement = "UPDATE MoviesPartiQBatch SET info =
'directors\":[\"Merian C. Cooper\", \"Ernest B. Schoedsack' where year=? and
title=?";

    List<AttributeValue> parametersRec1 = new ArrayList<>();

    // Update three records.
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2022"))

```

```
        .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 =
        BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();

        // Update record 2.
        List<AttributeValue> parametersRec2 = new ArrayList<>();
        AttributeValue attRec2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 =
        BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec2)
            .build();

        // Update record 3.
        List<AttributeValue> parametersRec3 = new ArrayList<>();
        AttributeValue attRec3 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec3a = AttributeValue.builder()
            .s("My Movie 3")
            .build();

        parametersRec3.add(attRec3);
```

```
        parametersRec3.add(attRec3a);
        BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec3)
            .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();
        myBatchStatementList.add(statementRequestRec1);
        myBatchStatementList.add(statementRequestRec2);
        myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();

        try {
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
            System.out.println("ExecuteStatement successful: " +
response.toString());
            System.out.println("Updated three movies using a batch
command.");
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }

    public static void deleteItemBatch(DynamoDbClient ddb) {
        String sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ?
and title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Specify three records to delete.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();
```

```
AttributeValue att2 = AttributeValue.builder()
    .s("My Movie 1")
    .build();

parametersRec1.add(att1);
parametersRec1.add(att2);

BatchStatementRequest statementRequestRec1 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec1)
    .build();

// Specify record 2.
List<AttributeValue> parametersRec2 = new ArrayList<>();
AttributeValue attRec2 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec2a = AttributeValue.builder()
    .s("My Movie 2")
    .build();

parametersRec2.add(attRec2);
parametersRec2.add(attRec2a);
BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec2)
    .build();

// Specify record 3.
List<AttributeValue> parametersRec3 = new ArrayList<>();
AttributeValue attRec3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec3a = AttributeValue.builder()
    .s("My Movie 3")
    .build();

parametersRec3.add(attRec3);
parametersRec3.add(attRec3a);
```

```
        BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
                        .statement(sqlStatement)
                        .parameters(parametersRec3)
                        .build();

        // Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();
myBatchStatementList.add(statementRequestRec1);
myBatchStatementList.add(statementRequestRec2);
myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
                        .statements(myBatchStatementList)
                        .build();

        try {
            ddb.batchExecuteStatement(batchRequest);
            System.out.println("Deleted three movies using a batch
command.");
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName)
    {
        DeleteTableRequest request = DeleteTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            ddb.deleteTable(request);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }
}
```

```
private static ExecuteStatementResponse
executeStatementRequest(DynamoDbClient db, String statement,
                        List<AttributeValue> parameters) {
    ExecuteStatementRequest request = ExecuteStatementRequest.builder()
        .statement(statement)
        .parameters(parameters)
        .build();

    return db.executeStatement(request);
}
}
```


- API の詳細については、「API リファレンス [BatchExecuteStatement](#)」の「」を参照してください。AWS SDK for Java 2.x

PartiQL を使用してテーブルに対してクエリを実行する

次のコードサンプルは、以下の操作方法を示しています。

- SELECT ステートメントを実行して項目を取得する。
- INSERT 文を実行して項目を追加する。
- UPDATE ステートメントを使用して項目を更新する。
- DELETE ステートメントを実行して項目を削除する。

SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public class ScenarioPartiQ {
    public static void main(String[] args) throws IOException {
        final String usage = ""
```

Usage:

```
        <fileName>

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

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String fileName = args[0];
    String tableName = "MoviesPartiQ";
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    System.out.println(
        "***** Creating an Amazon DynamoDB table named MoviesPartiQ with a
key named year and a sort key named title.");
    createTable(ddb, tableName);

    System.out.println("***** Loading data into the MoviesPartiQ table.");
    loadData(ddb, fileName);

    System.out.println("***** Getting data from the MoviesPartiQ table.");
    getItem(ddb);

    System.out.println("***** Putting a record into the MoviesPartiQ table.");
    putRecord(ddb);

    System.out.println("***** Updating a record.");
    updateTableItem(ddb);

    System.out.println("***** Querying the movies released in 2013.");
    queryTable(ddb);

    System.out.println("***** Deleting the Amazon DynamoDB table.");
    deleteDynamoDBTable(ddb, tableName);
    ddb.close();
}
```

```
public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("title")
        .attributeType("S")
        .build());

    ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
    KeySchemaElement key = KeySchemaElement.builder()
        .attributeName("year")
        .keyType(KeyType.HASH)
        .build();

    KeySchemaElement key2 = KeySchemaElement.builder()
        .attributeName("title")
        .keyType(KeyType.RANGE) // Sort
        .build();

    // Add KeySchemaElement objects to the list.
    tableKey.add(key);
    tableKey.add(key2);

    CreateTableRequest request = CreateTableRequest.builder()
        .keySchema(tableKey)
        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10))
            .build())
        .attributeDefinitions(attributeDefinitions)
        .tableName(tableName)
        .build();

    try {
        CreateTableResponse response = ddb.createTable(request);
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
```



```
        .build();

        // Wait until the Amazon DynamoDB table is created.
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitForTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        String newTable = response.tableDescription().tableName();
        System.out.println("The " + newTable + " was successfully created.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String fileName) throws
IOException {

    String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    List<AttributeValue> parameters = new ArrayList<>();
    while (iter.hasNext()) {

        // Add 200 movies to the table.
        if (t == 200)
            break;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf(year))
            .build();

        AttributeValue att2 = AttributeValue.builder()
```

```
        .s(title)
        .build();

    AttributeValue att3 = AttributeValue.builder()
        .s(info)
        .build();

    parameters.add(att1);
    parameters.add(att2);
    parameters.add(att3);

    // Insert the movie into the Amazon DynamoDB table.
    executeStatementRequest(ddb, sqlStatement, parameters);
    System.out.println("Added Movie " + title);

    parameters.remove(att1);
    parameters.remove(att2);
    parameters.remove(att3);
    t++;
}
}

public static void getItem(DynamoDbClient ddb) {

    String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n("2012")
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("The Perks of Being a Wallflower")
        .build();

    parameters.add(att1);
    parameters.add(att2);

    try {
        ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
        System.out.println("ExecuteStatement successful: " +
response.toString());

    } catch (DynamoDbException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void putRecord(DynamoDbClient ddb) {

    String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
    try {
        List<AttributeValue> parameters = new ArrayList<>();

        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2020"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie")
            .build();

        AttributeValue att3 = AttributeValue.builder()
            .s("No Information")
            .build();

        parameters.add(att1);
        parameters.add(att2);
        parameters.add(att3);

        executeStatementRequest(ddb, sqlStatement, parameters);
        System.out.println("Added new movie.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateTableItem(DynamoDbClient ddb) {

    String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian
C. Cooper\", \"Ernest B. Schoedsack' where year=? and title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2013"))
```

```
        .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("The East")
            .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            executeStatementRequest(ddb, sqlStatement, parameters);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }

    // Query the table where the year is 2013.
    public static void queryTable(DynamoDbClient ddb) {
        String sqlStatement = "SELECT * FROM MoviesPartiQ where year = ? ORDER BY
year";
        try {

            List<AttributeValue> parameters = new ArrayList<>();
            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2013"))
                .build();
            parameters.add(att1);

            // Get items in the table and write out the ID value.
            ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: " +
response.toString());

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
```

```
DeleteTableRequest request = DeleteTableRequest.builder()
    .tableName(tableName)
    .build();

try {
    ddb.deleteTable(request);

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println(tableName + " was successfully deleted!");
}

private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient
ddb, String statement,
    List<AttributeValue> parameters) {
    ExecuteStatementRequest request = ExecuteStatementRequest.builder()
        .statement(statement)
        .parameters(parameters)
        .build();

    return ddb.executeStatement(request);
}

private static void processResults(ExecuteStatementResponse
executeStatementResult) {
    System.out.println("ExecuteStatement successful: " +
executeStatementResult.toString());
}
}
```

- API の詳細については、「API リファレンス [ExecuteStatement](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## TTL 項目のクエリ

次のコード例は、TTL 項目をクエリする方法を示しています。

## SDK for Java 2.x

フィルタリングされた式をクエリして、DynamoDB テーブル内の TTL 項目を収集します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
import java.util.Optional;

// Get current time in epoch second format (comparing against expiry
attribute)
final long currentTime = System.currentTimeMillis() / 1000;

// A string that contains conditions that DynamoDB applies after the Query
operation, but before the data is returned to you.
final String keyConditionExpression = "#pk = :pk";

// The condition that specifies the key values for items to be retrieved by
the Query action.
final String filterExpression = "#ea > :ea";
final Map<String, String> expressionAttributeNames = ImmutableMap.of(
    "#pk", "primaryKey",
    "#ea", "expireAt");
final Map<String, AttributeValue> expressionAttributeValues =
ImmutableMap.of(
    ":pk", AttributeValue.builder().s(primaryKey).build(),
    ":ea",
AttributeValue.builder().s(String.valueOf(currentTime)).build()
);

final QueryRequest request = QueryRequest.builder()
    .tableName(tableName)
    .keyConditionExpression(keyConditionExpression)
    .filterExpression(filterExpression)
    .expressionAttributeNames(expressionAttributeNames)
    .expressionAttributeValues(expressionAttributeValues)
    .build();
```

```
try (DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build()) {
    final QueryResponse response = ddb.query(request);
    System.out.println(tableName + " Query operation with TTL successful.
Request id is "
        + response.responseMetadata().requestId());
    // Print the items that are not expired
    for (Map<String, AttributeValue> item : response.items()) {
        System.out.println(item.toString());
    }
} catch (ResourceNotFoundException e) {
    System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
    System.exit(1);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.exit(0);
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[Query](#)」を参照してください。

## 項目の TTL を更新する

次のコード例は、項目の TTL を更新する方法を示しています。

### SDK for Java 2.x

テーブル内の既存の DynamoDB 項目の TTL を更新します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
```

```
import java.util.Optional;

// Get current time in epoch second format
final long currentTime = System.currentTimeMillis() / 1000;
// Calculate expiration time 90 days from now in epoch second format
final long expireDate = currentTime + (90 * 24 * 60 * 60);
// An expression that defines one or more attributes to be updated, the
action to be performed on them, and new values for them.
final String updateExpression = "SET updatedAt=:c, expireAt=:e";

final ImmutableMap<String, AttributeValue> keyMap =
    ImmutableMap.of("primaryKey", AttributeValue.fromS(primaryKey),
        "sortKey", AttributeValue.fromS(sortKey));
final Map<String, AttributeValue> expressionAttributeValues =
ImmutableMap.of(
    ":c",
AttributeValue.builder().s(String.valueOf(currentTime)).build(),
    ":e", AttributeValue.builder().s(String.valueOf(expireDate)).build()
);

final UpdateItemRequest request = UpdateItemRequest.builder()
    .tableName(tableName)
    .key(keyMap)
    .updateExpression(updateExpression)
    .expressionAttributeValues(expressionAttributeValues)
    .build();
try (DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build()) {
    final UpdateItemResponse response = ddb.updateItem(request);
    System.out.println(tableName + " UpdateItem operation with TTL
successful. Request id is "
        + response.responseMetadata().requestId());
} catch (ResourceNotFoundException e) {
    System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
    System.exit(1);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.exit(0);
```



- API の詳細については、「API リファレンス [UpdateItem](#)」の「」を参照してください。AWS SDK for Java 2.x

## サーバーレスサンプル

### DynamoDB トリガーで Lambda 関数のバッチアイテムの失敗をレポートする

次のコード例は、DynamoDB ストリームからイベントを受信する Lambda 関数に部分的なバッチレスポンスを実装する方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [サーバーレスサンプル](#) リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Java を使用して Lambda で DynamoDB のバッチアイテム失敗のレポート。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
    Serializable> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
    {
```

```
        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
input.getRecords()) {
            try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
immediately.
                Lambda will immediately begin to retry processing from this
failed item onwards. */
                batchItemFailures.add(new
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

        return new StreamsEventResponse();
    }
}
```

## SDK for Java 2.x を使用する Amazon EC2 の例

次のコード例は、Amazon EC2 AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

## 開始方法

### Hello Amazon EC2

以下のコード例は、Amazon EC2 の利用開始方法を表示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
            .forEach(group -> System.out
                .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DescribeSecurityGroups](#)」の「」を参照してください。 AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### AllocateAddress

次の例は、AllocateAddress を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String allocateAddress(Ec2Client ec2) {
    try {
        AllocateAddressRequest allocateRequest =
        AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        AllocateAddressResponse allocateResponse =
        ec2.allocateAddress(allocateRequest);
        return allocateResponse.allocationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- API の詳細については、「API リファレンス [AllocateAddress](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## AssociateAddress

次の例は、AssociateAddress を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
    try {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
        return associateResponse.associationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- API の詳細については、「API リファレンス [AssociateAddress](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## AuthorizeSecurityGroupIngress

次の例は、AuthorizeSecurityGroupIngress を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupName)
```

```
        .ipPermissions(ipPerm, ipPerm2)
        .build();

        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.println("Successfully added ingress policy to security group
" + groupName);
        return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- API の詳細については、「API リファレンス [AuthorizeSecurityGroupIngress](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateKeyPair

次の例は、CreateKeyPair を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void createKeyPair(Ec2Client ec2, String keyName, String fileName)
{
    try {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();

        CreateKeyPairResponse response = ec2.createKeyPair(request);
        String content = response.keyMaterial();
        BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
```

```
        writer.write(content);
        writer.close();
        System.out.println("Successfully created key pair named " + keyName);

    } catch (Ec2Exception | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [CreateKeyPair](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateSecurityGroup

次の例は、CreateSecurityGroup を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
```



```
        .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupName)
            .ipPermissions(ipPerm, ipPerm2)
            .build();

        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.println("Successfully added ingress policy to security group
" + groupName);
        return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- API の詳細については、「API リファレンス [CreateSecurityGroup](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteKeyPair

次の例は、DeleteKeyPair を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named " + keyPair);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DeleteKeyPair](#)」の「」を参照してください。  
AWS SDK for Java 2.x

**DeleteSecurityGroup**

次の例は、DeleteSecurityGroup を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
```

```
try {
    DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
    .groupId(groupId)
    .build();

    ec2.deleteSecurityGroup(request);
    System.out.println("Successfully deleted security group with Id " +
groupId);

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス [DeleteSecurityGroup](#)」の「」を参照してください。AWS SDK for Java 2.x

## DescribeInstanceTypes

次の例は、DescribeInstanceTypes を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType;
    try {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
        .maxResults(10)
        .build();
```

```
DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
for (InstanceTypeInfo type : instanceTypes) {
    System.out.println("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
    System.out.println("Network information is " +
type.networkInfo().toString());
    System.out.println("Instance type is " +
type.instanceType().toString());
    instanceType = type.instanceType().toString();
    if (instanceType.compareTo("t2.2xlarge") == 0){
        return instanceType;
    }
}

} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
```

- API の詳細については、「API リファレンス [DescribeInstanceTypes](#)」の「」を参照してください。AWS SDK for Java 2.x

## DescribeInstances

次の例は、DescribeInstances を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
```

```
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeInstances {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        describeEC2Instances(ec2);
        ec2.close();
    }

    public static void describeEC2Instances(Ec2Client ec2) {
        try {
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                .maxResults(10)
                .build();

            DescribeInstancesIterable instancesIterable =
ec2.describeInstancesPaginator(request);
            instancesIterable.stream()
                .flatMap(r -> r.reservations().stream())
                .flatMap(reservation -> reservation.instances().stream())
                .forEach(instance -> {
                    System.out.println("Instance Id is " + instance.instanceId());
                    System.out.println("Image id is " + instance.imageId());
                    System.out.println("Instance type is " +
instance.instanceType());
                    System.out.println("Instance state name is " +
instance.state().name());
                    System.out.println("Monitoring information is " +
instance.monitoring().state());
                });
        }
    }
}
```

```
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorCode());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [DescribeInstances](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DescribeKeyPairs

次の例は、DescribeKeyPairs を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void describeKeys(Ec2Client ec2) {
    try {
        DescribeKeyPairsResponse response = ec2.describeKeyPairs();
        response.keyPairs().forEach(keyPair -> System.out.printf(
            "Found key pair with name %s " +
            "and fingerprint %s",
            keyPair.keyName(),
            keyPair.keyFingerprint()));
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DescribeKeyPairs](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DescribeSecurityGroups

次の例は、DescribeSecurityGroups を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
            .forEach(group -> System.out
                .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DescribeSecurityGroups](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DisassociateAddress

次の例は、DisassociateAddress を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DisassociateAddress](#)」の「」を参照してください。 AWS SDK for Java 2.x

## ReleaseAddress

次の例は、ReleaseAddress を使用する方法を説明しています。



## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void releaseEC2Address(Ec2Client ec2, String allocId) {
    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        ec2.releaseAddress(request);
        System.out.println("Successfully released Elastic IP address " +
allocId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [ReleaseAddress](#)」の「」を参照してください。AWS SDK for Java 2.x

**RunInstances**

次の例は、RunInstances を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This code example requires an AMI value. You can learn more about this value
 * by reading this documentation topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/AMIs.html
 */
public class CreateInstance {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <name> <amiId>

            Where:
                name - An instance name value that you can obtain from the AWS
Console (for example, ami-xxxxxx5c8b987b1a0).\s
                amiId - An Amazon Machine Image (AMI) value that you can obtain
from the AWS Console (for example, i-xxxxxx2734106d0ab).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String amiId = args[1];
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
```

```
        .region(region)
        .build();

String instanceId = createEC2Instance(ec2, name, amiId);
System.out.println("The Amazon EC2 Instance ID is " + instanceId);
ec2.close();
}

public static String createEC2Instance(Ec2Client ec2, String name, String amiId)
{
    RunInstancesRequest runRequest = RunInstancesRequest.builder()
        .imageId(amiId)
        .instanceType(InstanceType.T1_MICRO)
        .maxCount(1)
        .minCount(1)
        .build();

    // Use a waiter to wait until the instance is running.
    System.out.println("Going to start an EC2 instance using a waiter");
    RunInstancesResponse response = ec2.runInstances(runRequest);
    String instanceIdVal = response.instances().get(0).instanceId();
    ec2.waiter().waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal));
    Tag tag = Tag.builder()
        .key("Name")
        .value(name)
        .build();

    CreateTagsRequest tagRequest = CreateTagsRequest.builder()
        .resources(instanceIdVal)
        .tags(tag)
        .build();

    try {
        ec2.createTags(tagRequest);
        System.out.printf("Successfully started EC2 Instance %s based on AMI
%s", instanceIdVal, amiId);
        return instanceIdVal;
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

```
    }  
}
```

- API の詳細については、「API リファレンス [RunInstances](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## StartInstances

次の例は、StartInstances を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void startInstance(Ec2Client ec2, String instanceId) {  
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()  
        .overrideConfiguration(b -> b.maxAttempts(100))  
        .client(ec2)  
        .build();  
  
    StartInstancesRequest request = StartInstancesRequest.builder()  
        .instanceIds(instanceId)  
        .build();  
  
    System.out.println("Use an Ec2Waiter to wait for the instance to run. This  
will take a few minutes.");  
    ec2.startInstances(request);  
    DescribeInstancesRequest instanceRequest =  
DescribeInstancesRequest.builder()  
        .instanceIds(instanceId)  
        .build();  
  
    WaiterResponse<DescribeInstancesResponse> waiterResponse =  
ec2Waiter.waitUntilInstanceRunning(instanceRequest);  
    waiterResponse.matched().response().ifPresent(System.out::println);  
    System.out.println("Successfully started instance " + instanceId);  
}
```

```
}
```

- API の詳細については、「API リファレンス [StartInstances](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## StopInstances

次の例は、StopInstances を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();
    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully stopped instance " + instanceId);
}
```

- API の詳細については、「API リファレンス [StopInstances](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## TerminateInstances

次の例は、`TerminateInstances` を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void terminateEC2(Ec2Client ec2, String instanceId) {
    try {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
            .overrideConfiguration(b -> b.maxAttempts(100))
            .client(ec2)
            .build();

        TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to
terminate. This will take a few minutes.");
        ec2.terminateInstances(ti);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
            .waitUntilInstanceTerminated(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
        System.out.println(instanceId + " is terminated!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [TerminateInstances](#)」の「」を参照してください。AWS SDK for Java 2.x

## シナリオ

### レジリエントなサービスの構築と管理

次のコード例は、本、映画、曲のレコメンデーションを返す負荷分散型ウェブサービスの作成方法を示しています。この例は、障害に対するサービスの対応方法と、障害発生時の耐障害性を高めるためにサービスを再構築する方法を示しています。

- Amazon EC2 Auto Scaling グループを使用して、起動テンプレートに基づいて Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを作成し、インスタンス数を所定の範囲内に維持します。
- Elastic Load Balancing で HTTP リクエストを処理して配信します。
- Auto Scaling グループ内のインスタンスの状態を監視し、正常なインスタンスにのみリクエストを転送します。
- 各 EC2 インスタンスで Python ウェブサーバーを実行して HTTP リクエストを処理します。ウェブサーバーはレコメンデーションとヘルスチェックを返します。
- Amazon DynamoDB テーブルを使用してレコメンデーションサービスをシミュレートできます。
- AWS Systems Manager パラメータを更新して、リクエストとヘルスチェックに対するウェブサーバーの応答を制御します。

### SDK for Java 2.x

#### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
public class Main {

    public static final String fileName = "C:\\\\AWS\\resworkflow\\
\\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\\\AWS\\resworkflow\\
\\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\\\AWS\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\\\AWS\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
    {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
        System.out.println(DASHES);
    }
}
```



```
System.out.println(DASHES);
System.out.println("A - SETUP THE RESOURCES");
System.out.println("Press Enter when you're ready to start deploying
resources.");
in.nextLine();
deploy(loadBalancer);
System.out.println(DASHES);
System.out.println(DASHES);
System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
System.out.println("Press Enter when you're ready.");
in.nextLine();
demo(loadBalancer);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("C - DELETE THE RESOURCES");
System.out.println("""
    This concludes the demo of how to build and manage a resilient
service.

    To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
    that were created for this demo.
    """);

System.out.println("\n Do you want to delete the resources (y/n)? ");
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

if (userInput.equals("y")) {
    // Delete resources here
    deleteResources(loadBalancer, autoScaler, database);
    System.out.println("Resources deleted.");
} else {
    System.out.println("""
        Okay, we'll leave the resources intact.
        Don't forget to delete them when you're done with them or you
might incur unexpected charges.
        """);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The example has completed. ");
System.out.println("\n Thanks for watching!");
```

```
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
        throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScalingGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
            """
                For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
                to set up a load-balanced web service endpoint and explore
some ways to make it resilient
                against various kinds of failures.

                Some of the resources create by this demo are:
                \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
                \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
            """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
    }
}
```

```
System.out.println("Creating and populating a DynamoDB table named " +
tableName);
Database database = new Database();
database.createTable(tableName, fileName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
    This script starts a Python web server defined in the `server.py`
script. The web server
    listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
    For demo purposes, this server is run as the root user. In
production, the best practice is to
    run a web server, such as Apache, with least-privileged credentials.

    The template also defines an IAM policy that each instance uses to
assume a role that grants
    permissions to access the DynamoDB recommendation table and Systems
Manager parameters
    that control the flow of the demo.
    """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(
    "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
System.out.println("*** Wait 30 secs for the VPC to be created");
TimeUnit.SECONDS.sleep(30);
AutoScaler autoScaler = new AutoScaler();
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

System.out.println("""
    At this point, you have EC2 instances created. Once each instance
starts, it listens for
```

```
        HTTP requests. You can see these instances in the console or
continue with the demo.
        Press Enter when you're ready to continue.
        """);

    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating variables that control the flow of the demo.");
    ParameterHelper paramHelper = new ParameterHelper();
    paramHelper.reset();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("""
        Creating an Elastic Load Balancing target group and load balancer.
The target group
        defines how the load balancer connects to instances. The load
balancer provides a
        single endpoint where clients connect and dispatches requests to
instances in the group.
        """);

    String vpcId = autoScaler.getDefaultVPC();
    List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
    System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
    String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
    String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
    autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
    System.out.println("Verifying access to the load balancer endpoint...");
    boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
    if (!wasSuccessful) {
        System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to "http://checkip.amazonaws.com"
        HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
        try {
```

```
// Execute the request and get the response
HttpResponse response = httpClient.execute(httpGet);

// Read the response content.
String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

// Print the public IP address.
System.out.println("Public IP Address: " + ipAddress);
GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
if (!groupInfo.isPortOpen()) {
    System.out.println("""
        For this example to work, the default security group for
your default VPC must
        allow access from this computer. You can either add it
automatically from this
        example or add it yourself using the AWS Management
Console.
        """);

    System.out.println(
        "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
    String ans = in.nextLine();
    if ("y".equalsIgnoreCase(ans)) {
        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
        System.out.println("Security group rule added.");
    } else {
        System.out.println("No security group rule added.");
    }
}

} catch (AutoScalingException e) {
    e.printStackTrace();
}
} else if (wasSuccessful) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" + elbDnsName);
} else {
```

```
        System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
        System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
        System.out.println("you can successfully make a GET request to the load
balancer.");
    }

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();
    System.out.println("Read the ssm_only_policy.json file");
    String ssmOnlyPolicy = readFileAsString(ssmJSON);

    System.out.println("Resetting parameters to starting values for demo.");
    paramHelper.reset();

    System.out.println(
        """
                This part of the demonstration shows how to toggle
different parts of the system
                to create situations where the web service fails, and shows
how using a resilient
                architecture can keep the web service running in spite of
these failures.

                At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
        """);
    demoChoices(loadBalancer);

    System.out.println(
        """
                The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                The table name is contained in a Systems Manager parameter
named self.param_helper.table.
        """);
}
```

```
        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
        """);
    paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

    System.out.println(
        ""
        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
        """);
    demoChoices(loadBalancer);

    System.out.println(
        ""
        Instead of failing when the recommendation service fails,
the web service can return a static response.
        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
        """);
    paramHelper.put(paramHelper.failureResponse, "static");

    System.out.println("""
        Now, sending a GET request to the load balancer endpoint returns a
static response.
        The service still reports as healthy because health checks are still
shallow.
        """);
    demoChoices(loadBalancer);

    System.out.println("Let's reinstate the recommendation service.");
    paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

    System.out.println("""
        Let's also substitute bad credentials for one of the instances in
the target group so that it can't
        access the DynamoDB recommendation table. We will get an instance id
value.
        """);

    LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
    AutoScaler autoScaler = new AutoScaler();
```

```
// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " + profileAssociationId);
System.out.println("Replacing the profile for instance " + badInstanceId
+ " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
    """"
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
    """);

demoChoices(loadBalancer);

System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
    the web service can access the DynamoDB table that it depends on for
recommendations. Note that
    the deep health check is only for ELB routing and not for Auto
Scaling instance health.
    This kind of deep health check is not recommended for Auto Scaling
instance health, because it
    risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
    """);

System.out.println("""
    By implementing deep health checks, the load balancer can detect
when one of the instances is failing
    and take that instance out of rotation.
    """);

paramHelper.put(paramHelper.healthCheck, "deep");
```



```
        System.out.println("""
            Now, checking target health indicates that the instance with bad
credentials
            is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
            instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
            the load balancer takes unhealthy instances out of its rotation.
            """);

        demoChoices(loadBalancer);

        System.out.println(
            """
                Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                instance is to terminate it and let the auto scaler start a
new instance to replace it.
                """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
            Even while the instance is terminating and the new instance is
starting, sending a GET
            request to the web service continues to get a successful
recommendation response because
            the load balancer routes requests to the healthy instances. After
the replacement instance
            starts and reports as healthy, it is included in the load balancing
rotation.

            Note that terminating and replacing an instance typically takes
several minutes, during which time you
            can see the changing health check status until the new instance is
running and healthy.
            """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
```

```
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
    InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
```

```

        new
InputStreamReader(response.getEntity().getContent()));
        StringBuilder jsonResponse = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            jsonResponse.append(line);
        }
        reader.close();

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();
    }
    case 1 -> {
        System.out.println("\nChecking the health of load balancer
targets:\n");
        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
        Note that it can take a minute or two for the health
check to update
                after changes are made.
                """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value between
0-2. Please select again.");
}

} catch (java.util.InputMismatchException e) {

```

```
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}
```

Auto Scaling と Amazon EC2 のアクションをラップするクラスを作成します。

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
```

```
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
```

```
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
    .builder()
    .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                // name.
    .build();

// Replace the IAM instance profile association for the EC2 instance.
ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
    .builder()
    .iamInstanceProfile(iamInstanceProfile)
    .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
    .build();

try {
    getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
    // Handle the response as needed.
} catch (Ec2Exception e) {
    // Handle exceptions, log, or report the error.
    System.err.println("Error: " + e.getMessage());
}

System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
    newInstanceProfileName);
TimeUnit.SECONDS.sleep(15);
boolean instReady = false;
int tries = 0;

// Reboot after 60 seconds
while (!instReady) {
    if (tries % 6 == 0) {
        getEc2Client().rebootInstances(RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build());
        System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
    }
    tries++;
    try {
        TimeUnit.SECONDS.sleep(10);
    }
}
```

```

    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
    List<InstanceInformation> instanceInformationList =
informationResponse.getInstanceInformationList();
    for (InstanceInformation info : instanceInformationList) {
        if (info.getInstanceId().equals(instanceId)) {
            instReady = true;
            break;
        }
    }
}

SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
    .instanceIds(instanceId)
    .documentName("AWS-RunShellScript")
    .parameters(Collections.singletonMap("commands",
        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.");
}
}
```



## Elastic Load Balancing のアクションをラップするクラスを作成します。

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
        DescribeTargetGroupsRequest.builder()
            .names(targetGroupName)
            .build();

        DescribeTargetGroupsResponse tgResponse =
        getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
        DescribeTargetHealthRequest.builder()
            .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
            .build();

        DescribeTargetHealthResponse healthResponse =
        getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
```

```
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
```

```
boolean success = false;
int retries = 3;
CloseableHttpClient httpClient = HttpClients.createDefault();

// Create an HTTP GET request to the ELB.
HttpGet httpGet = new HttpGet("http://" + elbDnsName);
try {
    while ((!success) && (retries > 0)) {
        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);
        if (statusCode == 200) {
            success = true;
        } else {
            retries--;
            System.out.println("Got connection error from load balancer
endpoint, retrying...");
            TimeUnit.SECONDS.sleep(15);
        }
    }

} catch (org.apache.http.conn.HttpHostConnectException e) {
    System.out.println(e.getMessage());
}

System.out.println("Status.." + success);
return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
```

```
        .name(targetGroupName)
        .protocol(protocol)
        .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
                .scheme("internet-facing")
                .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
```

```
        .loadBalancerArns(lbARN)
        .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
```

DynamoDB を使用してレコメンデーションサービスをシミュレートするクラスを作成します。

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /**
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
```

```
* MediaType,
* forms a unique identifier for the recommended item.
*/
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
            .provisionedThroughput(
                ProvisionedThroughput.builder()
                    .readCapacityUnits(5L)
                    .writeCapacityUnits(5L)
                    .build())
            .build();

        getDynamoDbClient().createTable(createTableRequest);
        System.out.println("Creating table " + tableName + "...");

        // Wait until the Amazon DynamoDB table is created.
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();
    }
}
```

```
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Table " + tableName + " created.");

        // Add records to the table.
        populateTable(fileName, tableName);
    }
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
}
```



```
        System.out.println("Added all records to the " + tableName);
    }
}
```

Systems Manager のアクションをラップするクラスを作成します。

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [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](#)

## インスタンスを開始

次のコードサンプルは、以下の操作方法を示しています。

- キーペアとセキュリティグループを作成します。
- Amazon マシンイメージ (AMI) と互換性のあるインスタンスタイプを選択し、インスタンスを作成します。

Amazon EC2 インスタンスを停止し、再起動します。

- Elastic IP アドレスをインスタンスに関連付ける。
- SSH を使用してインスタンスに接続し、リソースをクリーンアップします。

## SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Creates an RSA key pair and saves the private key data as a .pem file.
 * 2. Lists key pairs.
 * 3. Creates a security group for the default VPC.
 * 4. Displays security group information.
 * 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 * 6. Gets more information about the image.
 * 7. Gets a list of instance types that are compatible with the selected AMI's
 * architecture.
 * 8. Creates an instance with the key pair, security group, AMI, and an
 * instance type.
 * 9. Displays information about the instance.
 * 10. Stops the instance and waits for it to stop.
 * 11. Starts the instance and waits for it to start.
 * 12. Allocates an Elastic IP address and associates it with the instance.
 * 13. Displays SSH connection info for the instance.
 * 14. Disassociates and deletes the Elastic IP address.
 * 15. Terminates the instance and waits for it to terminate.
 * 16. Deletes the security group.
 * 17. Deletes the key pair.
 */
```

```
public class EC2Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

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

        final String usage = ""

            Usage:
                <keyName> <fileName> <groupName> <groupDesc> <vpcId>

            Where:
                keyName - A key pair name (for example, TestKeyPair).\s
                fileName - A file name where the key information is written to.
\s
                groupName - The name of the security group.\s
                groupDesc - The description of the security group.\s
                vpcId - A VPC Id value. You can get this value from the AWS
Management Console.\s
                myIpAddress - The IP address of your development machine.\s

            """;

        if (args.length != 6) {
            System.out.println(usage);
            System.exit(1);
        }

        String keyName = args[0];
        String fileName = args[1];
        String groupName = args[2];
        String groupDesc = args[3];
        String vpcId = args[4];
        String myIpAddress = args[5];

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

        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
    }
}
```

```
System.out.println("Welcome to the Amazon EC2 example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an RSA key pair and save the private key
material as a .pem file.");
createKeyPair(ec2, keyName, fileName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. List key pairs.");
describeKeys(ec2);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Create a security group.");
String groupId = createSecurityGroup(ec2, groupName, groupDesc, vpcId,
myIpAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Display security group info for the newly created
security group.");
describeSecurityGroups(ec2, groupId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get a list of Amazon Linux 2 AMIs and selects one
with amzn2 in the name.");
String instanceId = getParaValues(ssmClient);
System.out.println("The instance Id is " + instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Get more information about an amzn2 image.");
String amiValue = describeImage(ec2, instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of instance types.");
String instanceType = getInstanceTypes(ec2);
System.out.println("The instance type is " + instanceType);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("8. Create an instance.");
String newInstanceId = runInstance(ec2, instanceType, keyName, groupName,
amiValue);
System.out.println("The instance Id is " + newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Display information about the running instance. ");
String ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Stop the instance and use a waiter.");
stopInstance(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Start the instance and use a waiter.");
startInstance(ec2, newInstanceId);
ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Allocate an Elastic IP address and associate it with
the instance.");
String allocationId = allocateAddress(ec2);
System.out.println("The allocation Id value is " + allocationId);
String associationId = associateAddress(ec2, newInstanceId, allocationId);
System.out.println("The associate Id value is " + associationId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Describe the instance again.");
ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
System.out.println("14. Disassociate and release the Elastic IP address.");
disassociateAddress(ec2, associationId);
releaseEC2Address(ec2, allocationId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Terminate the instance and use a waiter.");
terminateEC2(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("16. Delete the security group.");
deleteEC2SecGroup(ec2, groupId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("17. Delete the key.");
deleteKeys(ec2, keyName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("You successfully completed the Amazon EC2 scenario.");
System.out.println(DASHES);
ec2.close();
}

public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
    try {
        DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
        .groupId(groupId)
        .build();

        ec2.deleteSecurityGroup(request);
        System.out.println("Successfully deleted security group with Id " +
groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void terminateEC2(Ec2Client ec2, String instanceId) {
```

```
    try {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
            .overrideConfiguration(b -> b.maxAttempts(100))
            .client(ec2)
            .build();

        TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to
terminate. This will take a few minutes.");
        ec2.terminateInstances(ti);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
            .waitUntilInstanceTerminated(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
        System.out.println(instanceId + " is terminated!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named " + keyPair);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



```
public static void releaseEC2Address(Ec2Client ec2, String allocId) {
    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        ec2.releaseAddress(request);
        System.out.println("Successfully released Elastic IP address " +
allocId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
    try {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
        return associateResponse.associationId();
    }
}
```

```
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String allocateAddress(Ec2Client ec2) {
    try {
        AllocateAddressRequest allocateRequest =
AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        AllocateAddressResponse allocateResponse =
ec2.allocateAddress(allocateRequest);
        return allocateResponse.allocationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void startInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to run. This
will take a few minutes.");
    ec2.startInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();
```

```
        WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitForInstanceRunning(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
    }

    public static void stopInstance(Ec2Client ec2, String instanceId) {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
            .overrideConfiguration(b -> b.maxAttempts(100))
            .client(ec2)
            .build();
        StopInstancesRequest request = StopInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
        ec2.stopInstances(request);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitForInstanceStopped(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully stopped instance " + instanceId);
    }

    public static String describeEC2Instances(Ec2Client ec2, String newInstanceId) {
        try {
            String pubAddress = "";
            boolean isRunning = false;
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                .instanceIds(newInstanceId)
                .build();

            while (!isRunning) {
                DescribeInstancesResponse response = ec2.describeInstances(request);
                String state =
response.reservations().get(0).instances().get(0).state().name().name();
                if (state.compareTo("RUNNING") == 0) {
                    System.out.println("Image id is " +
response.reservations().get(0).instances().get(0).imageId());
                }
            }
        }
    }
}
```

```
        System.out.println(
            "Instance type is " +
response.reservations().get(0).instances().get(0).instanceType());
        System.out.println(
            "Instance state is " +
response.reservations().get(0).instances().get(0).state().name());
        pubAddress =
response.reservations().get(0).instances().get(0).publicIpAddress();
        System.out.println("Instance address is " + pubAddress);
        isRunning = true;
    }
}
return pubAddress;
} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}

public static String runInstance(Ec2Client ec2, String instanceType, String
keyName, String groupName,
    String amiId) {
    try {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .instanceType(instanceType)
            .keyName(keyName)
            .securityGroups(groupName)
            .maxCount(1)
            .minCount(1)
            .imageId(amiId)
            .build();

        System.out.println("Going to start an EC2 instance using a waiter");
        RunInstancesResponse response = ec2.runInstances(runRequest);
        String instanceIdVal = response.instances().get(0).instanceId();
        ec2.waiter().waitUntilInstanceRunning(r ->
r.instanceIds(instanceIdVal));
        System.out.println("Successfully started EC2 instance " + instanceIdVal
+ " based on AMI " + amiId);
        return instanceIdVal;

    } catch (SsmException e) {
        System.err.println(e.getMessage());
    }
}
```

```
        System.exit(1);
    }
    return "";
}

// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType;
    try {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .maxResults(10)
            .build();

        DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
        List<InstanceTypeInfo> instanceTypes = response.getInstanceTypes();
        for (InstanceTypeInfo type : instanceTypes) {
            System.out.println("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
            System.out.println("Network information is " +
type.networkInfo().toString());
            System.out.println("Instance type is " +
type.getInstanceType().toString());
            instanceType = type.getInstanceType().toString();
            if (instanceType.compareTo("t2.2xlarge") == 0){
                return instanceType;
            }
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Display the Description field that corresponds to the instance Id value.
public static String describeImage(Ec2Client ec2, String instanceId) {
    try {
        DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
            .imageIds(instanceId)
            .build();
```

```
        DescribeImagesResponse response = ec2.describeImages(imagesRequest);
        System.out.println("The description of the first image is " +
response.images().get(0).description());
        System.out.println("The name of the first image is " +
response.images().get(0).name());

        // Return the image Id value.
        return response.images().get(0).imageId();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Get the Id value of an instance with amzn2 in the name.
public static String getParaValues(SsmClient ssmClient) {
    try {
        GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
            .path("/aws/service/ami-amazon-linux-latest")
            .build();

        GetParametersByPathIterable responses =
ssmClient.getParametersByPathPaginator(parameterRequest);
        for
(ssoftware.amazon.awssdk.services.ssm.model.GetParametersByPathResponse response :
responses) {
            System.out.println("Test " + response.nextToken());
            List<Parameter> parameterList = response.parameters();
            for (Parameter para : parameterList) {
                System.out.println("The name of the para is: " + para.name());
                System.out.println("The type of the para is: " + para.type());
                if (filterName(para.name())) {
                    return para.value();
                }
            }
        }
    }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
        return "";
    }

    // Return true if the name has amzn2 in it. For example:
    // /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-arm64-gp2
    private static boolean filterName(String name) {
        String[] parts = name.split("/");
        String myValue = parts[4];
        return myValue.contains("amzn2");
    }

    public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
        try {
            DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
                .groupIds(groupId)
                .build();

            // Use a paginator.
            DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
            listGroups.stream()
                .flatMap(r -> r.securityGroups().stream())
                .forEach(group -> System.out
                    .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
        String myIpAddress) {
        try {
            CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
                .groupName(groupName)
                .description(groupDesc)
                .vpcId(vpcId)
                .build();
```

```
        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupName)
            .ipPermissions(ipPerm, ipPerm2)
            .build();

        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.println("Successfully added ingress policy to security group
" + groupName);
        return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void describeKeys(Ec2Client ec2) {
    try {
        DescribeKeyPairsResponse response = ec2.describeKeyPairs();
        response.keyPairs().forEach(keyPair -> System.out.printf(
            "Found key pair with name %s " +
            "and fingerprint %s",
```



```
        keyPair.keyName(),
        keyPair.keyFingerprint()));

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createKeyPair(Ec2Client ec2, String keyName, String fileName)
{
    try {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();

        CreateKeyPairResponse response = ec2.createKeyPair(request);
        String content = response.keyMaterial();
        BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
        writer.write(content);
        writer.close();
        System.out.println("Successfully created key pair named " + keyName);

    } catch (Ec2Exception | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [AllocateAddress](#)
  - [AssociateAddress](#)
  - [AuthorizeSecurityGroupIngress](#)
  - [CreateKeyPair](#)
  - [CreateSecurityGroup](#)
  - [DeleteKeyPair](#)
  - [DeleteSecurityGroup](#)

- [DescribeImages](#)
- [DescribeInstanceTypes](#)
- [DescribeInstances](#)
- [DescribeKeyPairs](#)
- [DescribeSecurityGroups](#)
- [DisassociateAddress](#)
- [ReleaseAddress](#)
- [RunInstances](#)
- [StartInstances](#)
- [StopInstances](#)
- [TerminateInstances](#)
- [UnmonitorInstances](#)

## SDK for Java 2.x を使用する Amazon ECS の例

次のコード例は、Amazon ECS AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- [アクション](#)

### アクション

#### CreateCluster

次の例は、CreateCluster を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
}
}
```

- API の詳細については、「API リファレンス [CreateCluster](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateService

次の例は、CreateService を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
}
}
```

- API の詳細については、「API リファレンス [CreateService](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteService

次の例は、DeleteService を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
}
}
```

- API の詳細については、「API リファレンス [DeleteService](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DescribeClusters

次の例は、DescribeClusters を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeClustersRequest;
import software.amazon.awssdk.services.ecs.model.DescribeClustersResponse;
import software.amazon.awssdk.services.ecs.model.Cluster;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DescribeClusters {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <clusterArn> \s

            Where:
            clusterArn - The ARN of the ECS cluster to describe.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterArn = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        descCluster(ecsClient, clusterArn);
    }

    public static void descCluster(EcsClient ecsClient, String clusterArn) {
        try {
            DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
                .clusters(clusterArn)
                .build();

            DescribeClustersResponse response =
ecsClient.describeClusters(clustersRequest);
            List<Cluster> clusters = response.clusters();
            for (Cluster cluster : clusters) {
                System.out.println("The cluster name is " + cluster.clusterName());
            }
        } catch (EcsException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [DescribeClusters](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DescribeTasks

次の例は、DescribeTasks を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

```
}
```

- API の詳細については、「API リファレンス [DescribeTasks](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListClusters

次の例は、ListClusters を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「API リファレンス [ListClusters](#)」の「」を参照してください。AWS SDK for Java 2.x

## UpdateService

次の例は、UpdateService を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [UpdateService](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## Elastic Load Balancing - SDK for Java 2.x を使用したバージョン 2 の例

次のコード例は、Elastic Load Balancing - バージョン 2 AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

### 開始方法

#### Hello Elastic Load Balancing

次のコード例は、Elastic Load Balancing の使用を開始する方法を示しています。



## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

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

- API の詳細については、「API リファレンス [DescribeLoadBalancers](#)」の「」を参照してください。 AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

**CreateListener**

次の例は、CreateListener を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/*
 * Creates an Elastic Load Balancing load balancer that uses the specified
 * subnets
 * and forwards requests to the specified target group.
 */
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
    }
}
```

```
WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
    .waitUntilLoadBalancerAvailable(request);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Load Balancer " + lbName + " is available.");

// Get the DNS name (endpoint) of the load balancer.
String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

// Create a listener for the load balance.
Action action = Action.builder()
    .targetGroupArn(targetGroupARN)
    .type("forward")
    .build();

CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
    .defaultActions(action)
    .port(port)
    .protocol(protocol)
    .defaultActions(action)
    .build();

getLoadBalancerClient().createListener(listenerRequest);
System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
    + targetGroupARN);

// Return the load balancer DNS name.
return lbDNSName;

} catch (ElasticLoadBalancingV2Exception e) {
    e.printStackTrace();
}
return "";
}
```

- API の詳細については、「API リファレンス [CreateListener](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateLoadBalancer

次の例は、CreateLoadBalancer を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/*
 * Creates an Elastic Load Balancing load balancer that uses the specified
 * subnets
 * and forwards requests to the specified target group.
 */
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();
```

```
        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
```

- API の詳細については、「API リファレンス [CreateLoadBalancer](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateTargetGroup

次の例は、CreateTargetGroup を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/*
 * 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;
}
```

```
}
```

- API の詳細については、「API リファレンス [CreateTargetGroup](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteLoadBalancer

次の例は、DeleteLoadBalancer を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

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

- API の詳細については、「API リファレンス [DeleteLoadBalancer](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteTargetGroup

次の例は、DeleteTargetGroup を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

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

- API の詳細については、「API リファレンス [DeleteTargetGroup](#)」の「」を参照してください。AWS SDK for Java 2.x



## DescribeTargetHealth

次の例は、DescribeTargetHealth を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

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

- API の詳細については、「API リファレンス [DescribeTargetHealth](#)」の「」を参照してください。 AWS SDK for Java 2.x

## シナリオ

### レジリエントなサービスの構築と管理

次のコード例は、本、映画、曲のレコメンデーションを返す負荷分散型ウェブサービスの作成方法を示しています。この例は、障害に対するサービスの対応方法と、障害発生時の耐障害性を高めるためにサービスを再構築する方法を示しています。

- Amazon EC2 Auto Scaling グループを使用して、起動テンプレートに基づいて Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを作成し、インスタンス数を所定の範囲内に維持します。
- Elastic Load Balancing で HTTP リクエストを処理して配信します。
- Auto Scaling グループ内のインスタンスの状態を監視し、正常なインスタンスにのみリクエストを転送します。
- 各 EC2 インスタンスで Python ウェブサーバーを実行して HTTP リクエストを処理します。ウェブサーバーはレコメンデーションとヘルスチェックを返します。
- Amazon DynamoDB テーブルを使用してレコメンデーションサービスをシミュレートできます。
- AWS Systems Manager パラメータを更新して、リクエストとヘルスチェックに対するウェブサーバーの応答を制御します。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
public class Main {  
  
    public static final String fileName = "C:\\AWS\\resworkflow\\  
\\recommendations.json"; // Modify file location.  
    public static final String tableName = "doc-example-recommendation-service";  
    public static final String startScript = "C:\\AWS\\resworkflow\\  
\\server_startup_script.sh"; // Modify file location.  
}
```

```
public static final String policyFile = "C:\\AWS\\resworkflow\\
\\instance_policy.json"; // Modify file location.
public static final String ssmJSON = "C:\\AWS\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
public static final String templateName = "doc-example-resilience-template";
public static final String roleName = "doc-example-resilience-role";
public static final String policyName = "doc-example-resilience-pol";
public static final String profileName = "doc-example-resilience-prof";

public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

public static final String targetGroupName = "doc-example-resilience-tg";
public static final String autoScalingGroupName = "doc-example-resilience-
group";
public static final String lbName = "doc-example-resilience-lb";
public static final String protocol = "HTTP";
public static final int port = 80;

public static final String DASHES = new String(new char[80]).replace("\\0", "-");

public static void main(String[] args) throws IOException, InterruptedException
{
    Scanner in = new Scanner(System.in);
    Database database = new Database();
    AutoScaler autoScaler = new AutoScaler();
    LoadBalancer loadBalancer = new LoadBalancer();

    System.out.println(DASHES);
    System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("A - SETUP THE RESOURCES");
    System.out.println("Press Enter when you're ready to start deploying
resources.");
    in.nextLine();
    deploy(loadBalancer);
    System.out.println(DASHES);
}
```

```
System.out.println(DASHES);
System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
System.out.println("Press Enter when you're ready.");
in.nextLine();
demo(loadBalancer);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("C - DELETE THE RESOURCES");
System.out.println("""
    This concludes the demo of how to build and manage a resilient
service.

    To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
    that were created for this demo.
    """);

System.out.println("\n Do you want to delete the resources (y/n)? ");
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

if (userInput.equals("y")) {
    // Delete resources here
    deleteResources(loadBalancer, autoScaler, database);
    System.out.println("Resources deleted.");
} else {
    System.out.println("""
        Okay, we'll leave the resources intact.
        Don't forget to delete them when you're done with them or you
might incur unexpected charges.
    """);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The example has completed. ");
System.out.println("\n Thanks for watching!");
System.out.println(DASHES);
}

// Deletes the AWS resources used in this example.
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
    throws IOException, InterruptedException {
    loadBalancer.deleteLoadBalancer(lbName);
```

```
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScalingGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
    InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
            """
                For this demo, we'll use the AWS SDK for Java (v2) to create
    several AWS resources
                to set up a load-balanced web service endpoint and explore
    some ways to make it resilient
                against various kinds of failures.

                Some of the resources create by this demo are:
                \t* A DynamoDB table that the web service depends on to
    provide book, movie, and song recommendations.
                \t* An EC2 launch template that defines EC2 instances that
    each contain a Python web server.
                \t* An EC2 Auto Scaling group that manages EC2 instances
    across several Availability Zones.
                \t* An Elastic Load Balancing (ELB) load balancer that
    targets the Auto Scaling group to distribute requests.
            """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
    tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
```

Creating an EC2 launch template that runs '{startup\_script}' when an instance starts.

This script starts a Python web server defined in the `server.py` script. The web server

listens to HTTP requests on port 80 and responds to requests to '/' and to '/healthcheck'.

For demo purposes, this server is run as the root user. In production, the best practice is to run a web server, such as Apache, with least-privileged credentials.

The template also defines an IAM policy that each instance uses to assume a role that grants permissions to access the DynamoDB recommendation table and Systems Manager parameters that control the flow of the demo.

```
""");
```

```
LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println(
    "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
System.out.println("*** Wait 30 secs for the VPC to be created");
TimeUnit.SECONDS.sleep(30);
AutoScaler autoScaler = new AutoScaler();
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);
```

```
System.out.println("""
    At this point, you have EC2 instances created. Once each instance
starts, it listens for
    HTTP requests. You can see these instances in the console or
continue with the demo.
    Press Enter when you're ready to continue.
""");
```

```
in.nextLine();
System.out.println(DASHES);
```

```
System.out.println(DASHES);
```

```
System.out.println("Creating variables that control the flow of the demo.");
ParameterHelper paramHelper = new ParameterHelper();
paramHelper.reset();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an Elastic Load Balancing target group and load balancer.
The target group
    defines how the load balancer connects to instances. The load
balancer provides a
    single endpoint where clients connect and dispatches requests to
instances in the group.
    """);

String vpcId = autoScaler.getDefaultVPC();
List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
System.out.println("Verifying access to the load balancer endpoint...");
boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
if (!wasSuccessful) {
    System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to "http://checkip.amazonaws.com"
   HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
    try {
        // Execute the request and get the response
        HttpResponse response = httpClient.execute(httpGet);

        // Read the response content.
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

        // Print the public IP address.
        System.out.println("Public IP Address: " + ipAddress);
```

```
        GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
        if (!groupInfo.isPortOpen()) {
            System.out.println("""
                For this example to work, the default security group for
your default VPC must
                allow access from this computer. You can either add it
automatically from this
                example or add it yourself using the AWS Management
Console.
                """);

            System.out.println(
                "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
            System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
            String ans = in.nextLine();
            if ("y".equalsIgnoreCase(ans)) {
                autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                System.out.println("Security group rule added.");
            } else {
                System.out.println("No security group rule added.");
            }
        }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
} else if (wasSuccessful) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" + elbDnsName);
} else {
    System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
    System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
    System.out.println("you can successfully make a GET request to the load
balancer.");
}
```



```
        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
            """
                This part of the demonstration shows how to toggle
different parts of the system
                to create situations where the web service fails, and shows
how using a resilient
                architecture can keep the web service running in spite of
these failures.

                At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
            """);
        demoChoices(loadBalancer);

        System.out.println(
            """
                The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
            """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
            """
                \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
```

```
        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
        """);
demoChoices(loadBalancer);

System.out.println(
    ""
        Instead of failing when the recommendation service fails,
the web service can return a static response.
        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
        """);
paramHelper.put(paramHelper.failureResponse, "static");

System.out.println("""
    Now, sending a GET request to the load balancer endpoint returns a
static response.
    The service still reports as healthy because health checks are still
shallow.
        """);
demoChoices(loadBalancer);

System.out.println("Let's reinstate the recommendation service.");
paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

System.out.println("""
    Let's also substitute bad credentials for one of the instances in
the target group so that it can't
    access the DynamoDB recommendation table. We will get an instance id
value.
        """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
AutoScaler autoScaler = new AutoScaler();

// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " + profileAssociationId);
```

```
System.out.println("Replacing the profile for instance " + badInstanceId
    + " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
    ""
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
    "");

demoChoices(loadBalancer);

System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
        the web service can access the DynamoDB table that it depends on for
recommendations. Note that
        the deep health check is only for ELB routing and not for Auto
Scaling instance health.
        This kind of deep health check is not recommended for Auto Scaling
instance health, because it
        risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
    """);

System.out.println("""
    By implementing deep health checks, the load balancer can detect
when one of the instances is failing
        and take that instance out of rotation.
    """);

paramHelper.put(paramHelper.healthCheck, "deep");

System.out.println("""
    Now, checking target health indicates that the instance with bad
credentials
        is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
        instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
        the load balancer takes unhealthy instances out of its rotation.
    """);
```

```
        """);

        demoChoices(loadBalancer);

        System.out.println(
            ""
                "Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                instance is to terminate it and let the auto scaler start a
new instance to replace it.
            """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
            Even while the instance is terminating and the new instance is
starting, sending a GET
            request to the web service continues to get a successful
recommendation response because
            the load balancer routes requests to the healthy instances. After
the replacement instance
            starts and reports as healthy, it is included in the load balancing
rotation.

            Note that terminating and replacing an instance typically takes
several minutes, during which time you
            can see the changing health check status until the new instance is
running and healthy.
        """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        }
    }
}
```

```
};
Scanner scanner = new Scanner(System.in);

while (true) {
    System.out.println("-".repeat(88));
    System.out.println("See the current state of the service by selecting
one of the following choices:");
    for (int i = 0; i < actions.length; i++) {
        System.out.println(i + ": " + actions[i]);
    }

    try {
        System.out.print("\nWhich action would you like to take? ");
        int choice = scanner.nextInt();
        System.out.println("-".repeat(88));

        switch (choice) {
            case 0 -> {
                System.out.println("Request:\n");
                System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                CloseableHttpClient httpClient =
HttpClientClients.createDefault();

                // Create an HTTP GET request to the ELB.
                HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);

                // Display the JSON response
                BufferedReader reader = new BufferedReader(
                    new
InputStreamReader(response.getEntity().getContent()));
                StringBuilder jsonResponse = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null) {
                    jsonResponse.append(line);
                }
                reader.close();
            }
        }
    }
}
```

```

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();

    }
    case 1 -> {
        System.out.println("\nChecking the health of load balancer
targets:\n");

        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value between
0-2. Please select again.");
}

} catch (java.util.InputMismatchException e) {
    System.out.println("Invalid input. Please select again.");
    scanner.nextLine(); // Clear the input buffer.
}
}
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}

```

```
}  
}
```

Auto Scaling と Amazon EC2 のアクションをラップするクラスを作成します。

```
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.getInstanceProfile().getInstanceProfileName();
        System.out.println(name);

        RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .roleName(roleName)
            .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
            .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(attachedPolicy.policyArn())
                .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");
```

```
        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /**
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                .name("group-name")
                .values("default")

```

```
        .build();

    Filter filter1 = Filter.builder()
        .name("vpc-id")
        .values(VPC)
        .build();

    DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
        .filters(filter, filter1)
        .build();

    DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
        .describeSecurityGroups(securityGroupsRequest);
    String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
    groupInfo.setGroupName(securityGroup);

    for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
        System.out.println("Found security group: " + secGroup.groupId());

        for (IpPermission ipPermission : secGroup.ipPermissions()) {
            if (ipPermission.fromPort() == port) {
                System.out.println("Found inbound rule: " + ipPermission);
                for (IpRange ipRange : ipPermission.ipRanges()) {
                    String cidrIp = ipRange.cidrIp();
                    if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                        System.out.println(cidrIp + " is applicable");
                        portIsOpen = true;
                    }
                }

                if (!ipPermission.prefixListIds().isEmpty()) {
                    System.out.println("Prefix lList is applicable");
                    portIsOpen = true;
                }

                if (!portIsOpen) {
                    System.out
                        .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                }
            }
        }
    }
}
```

```
        } else {
            break;
        }
    }
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/*
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {
```

```
// Get availability zones.
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
    .builder()
    .build();

DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

String availabilityZones = String.join(",", availabilityZoneNames);
LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

String[] zones = availabilityZones.split(",");
CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

try {
    getAutoScalingClient().createAutoScalingGroup(groupRequest);

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
return zones;
}
```

```
public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();

    DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
        .filters(vpcFilter, azFilter, defaultForAZ)
        .build();

    DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
    subnets = response.subnets();
    return subnets;
}
```



```
// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
```

```
ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
for (Policy policy : listPoliciesResponse.policies()) {
    if (policy.policyName().equals(policyName)) {
        // List the entities (users, groups, roles) that are attached to the
policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
    .builder()
    .policyArn(policy.arn())
    .build();
ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
    .listEntitiesForPolicy(listEntitiesRequest);
if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
    || !listEntitiesResponse.policyRoles().isEmpty()) {
    // Detach the policy from any entities it is attached to.
DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
    .policyArn(policy.arn())
    .roleName(roleName) // Specify the name of the IAM role
    .build();

    getIAMClient().detachRolePolicy(detachPolicyRequest);
    System.out.println("Policy detached from entities.");
}

// Now, you can delete the policy.
DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
    .policyArn(policy.arn())
    .build();

getIAMClient().deletePolicy(deletePolicyRequest);
System.out.println("Policy deleted successfully.");
break;
}
}

// List the roles associated with the instance profile
```

```
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .roleName(roleName) // Remove the extra dot here
        .build();

    getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
    System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
}

// Delete the instance profile after removing all roles
DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
    .build();

getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
System.out.println(InstanceProfile + " Deleted");
System.out.println("All roles and policies are deleted.");
}
}
```

Elastic Load Balancing のアクションをラップするクラスを作成します。

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
```

```
        .region(Region.US_EAST_1)
        .build();
    }

    return elasticLoadBalancingV2Client;
}

// Checks the health of the instances in the target group.
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
        .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
```

```

        .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
        .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {

```

```
        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);
        if (statusCode == 200) {
            success = true;
        } else {
            retries--;
            System.out.println("Got connection error from load balancer
endpoint, retrying...");
            TimeUnit.SECONDS.sleep(15);
        }
    }

} catch (org.apache.http.conn.HttpHostConnectException e) {
    System.out.println(e.getMessage());
}

System.out.println("Status.." + success);
return success;
}

/**
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
}
```

```
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
                .scheme("internet-facing")
                .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                .loadBalancerArns(lbARN)
                .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                .waitUntilLoadBalancerAvailable(request);
```

```
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Load Balancer " + lbName + " is available.");

// Get the DNS name (endpoint) of the load balancer.
String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

// Create a listener for the load balance.
Action action = Action.builder()
    .targetGroupArn(targetGroupARN)
    .type("forward")
    .build();

CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
    .defaultActions(action)
    .port(port)
    .protocol(protocol)
    .defaultActions(action)
    .build();

getLoadBalancerClient().createListener(listenerRequest);
System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
    + targetGroupARN);

// Return the load balancer DNS name.
return lbDNSName;

} catch (ElasticLoadBalancingV2Exception e) {
    e.printStackTrace();
}
return "";
}
}
```

DynamoDB を使用してレコメンデーションサービスをシミュレートするクラスを作成します。

```
public class Database {

    private static DynamoDbClient dynamoDbClient;
```



```
public static DynamoDbClient getDynamoDbClient() {
    if (dynamoDbClient == null) {
        dynamoDbClient = DynamoDbClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return dynamoDbClient;
}

// Checks to see if the Amazon DynamoDB table exists.
private boolean doesTableExist(String tableName) {
    try {
        // Describe the table and catch any exceptions.
        DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        getDynamoDbClient().describeTable(describeTableRequest);
        System.out.println("Table '" + tableName + "' exists.");
        return true;
    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " + e.getMessage());
    }
    return false;
}

/**
 * Creates a DynamoDB table to use a recommendation service. The table has a
 * hash key named 'MediaType' that defines the type of media recommended, such
 * as
 * Book or Movie, and a range key named 'ItemId' that, combined with the
 * MediaType,
 * forms a unique identifier for the recommended item.
 */
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
    }
}
```

```
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
            .provisionedThroughput(
                ProvisionedThroughput.builder()
                    .readCapacityUnits(5L)
                    .writeCapacityUnits(5L)
                    .build())
            .build();

        getDynamoDbClient().createTable(createTableRequest);
        System.out.println("Creating table " + tableName + "...");

        // Wait until the Amazon DynamoDB table is created.
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Table " + tableName + " created.");

        // Add records to the table.
        populateTable(fileName, tableName);
    }
}
```

```
public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
    System.out.println("Added all records to the " + tableName);
}
}
```

Systems Manager のアクションをラップするクラスを作成します。

```
public class ParameterHelper {
```

```
String tableName = "doc-example-resilient-architecture-table";
String dyntable = "doc-example-recommendation-service";
String failureResponse = "doc-example-resilient-architecture-failure-response";
String healthCheck = "doc-example-resilient-architecture-health-check";

public void reset() {
    put(dyntable, tableName);
    put(failureResponse, "none");
    put(healthCheck, "shallow");
}

public void put(String name, String value) {
    SsmClient ssmClient = SsmClient.builder()
        .region(Region.US_EAST_1)
        .build();

    PutParameterRequest parameterRequest = PutParameterRequest.builder()
        .name(name)
        .value(value)
        .overwrite(true)
        .type("String")
        .build();

    ssmClient.putParameter(parameterRequest);
    System.out.printf("Setting demo parameter %s to '%s'.", name, value);
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [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 SDK for Java 2.x を使用した の例

次のコード例は、AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています MediaStore。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

## • [アクション](#)

### アクション

#### CreateContainer

次の例は、CreateContainer を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <containerName>

            Where:
                containerName - The name of the container to create.
        """;
    }
}
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }

    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
        String containerName) {
        try {
            CreateContainerRequest containerRequest =
                CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
                mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
                    containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
                containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [CreateContainer](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteContainer

次の例は、DeleteContainer を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <containerName>

            Where:
                containerName - The name of the container to create.
    }
}
```



```
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String containerName = args[0];
    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();

    createMediaContainer(mediaStoreClient, containerName);
    mediaStoreClient.close();
}

public static void createMediaContainer(MediaStoreClient mediaStoreClient,
String containerName) {
    try {
        CreateContainerRequest containerRequest =
CreateContainerRequest.builder()
            .containerName(containerName)
            .build();

        CreateContainerResponse containerResponse =
mediaStoreClient.createContainer(containerRequest);
        String status = containerResponse.container().status().toString();
        while (!status.equalsIgnoreCase("Active")) {
            status = DescribeContainer.checkContainer(mediaStoreClient,
containerName);
            System.out.println("Status - " + status);
            Thread.sleep(sleepTime * 1000);
        }

        System.out.println("The container ARN value is " +
containerResponse.container().arn());
        System.out.println("Finished ");

    } catch (MediaStoreException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}
```

- API の詳細については、「API リファレンス [DeleteContainer](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteObject

次の例は、DeleteObject を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.DeleteObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

        Usage:    <completePath> <containerName>
```

```
        Where:
            completePath - The path (including the container) of the item to
delete.
            containerName - The name of the container.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String completePath = args[0];
    String containerName = args[1];
    Region region = Region.US_EAST_1;
    URI uri = new URI(getEndpoint(containerName));

    MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
        .endpointOverride(uri)
        .region(region)
        .build();

    deleteMediaObject(mediaStoreData, completePath);
    mediaStoreData.close();
}

public static void deleteMediaObject(MediaStoreDataClient mediaStoreData, String
completePath) {
    try {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
            .path(completePath)
            .build();

        mediaStoreData.deleteObject(deleteObjectRequest);

    } catch (MediaStoreDataException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

private static String getEndpoint(String containerName) {
    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
```

```
        .region(region)
        .build();

    DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
        .containerName(containerName)
        .build();

    DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
    mediaStoreClient.close();
    return response.container().endpoint();
}
}
```

- API の詳細については、「API リファレンス [DeleteObject](#)」の「」を参照してください。AWS SDK for Java 2.x

## DescribeContainer

次の例は、DescribeContainer を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DescribeContainer {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <containerName>

            Where:
                containerName - The name of the container to describe.
            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        System.out.println("Status is " + checkContainer(mediaStoreClient,
containerName));
        mediaStoreClient.close();
    }

    public static String checkContainer(MediaStoreClient mediaStoreClient, String
containerName) {
        try {
            DescribeContainerRequest describeContainerRequest =
DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

            DescribeContainerResponse containerResponse =
mediaStoreClient.describeContainer(describeContainerRequest);
            System.out.println("The container name is " +
containerResponse.container().name());
        }
    }
}
```

```
        System.out.println("The container ARN is " +
containerResponse.container().arn());
        return containerResponse.container().status().toString();

    } catch (MediaStoreException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- API の詳細については、「API リファレンス [DescribeContainer](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## GetObject

次の例は、GetObject を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectResponse;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
```

```
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

            Usage:    <completePath> <containerName> <savePath>

            Where:
                completePath - The path of the object in the container (for
example, Videos5/sampleVideo.mp4).
                containerName - The name of the container.
                savePath - The path on the local drive where the file is saved,
including the file name (for example, C:/AWS/myvid.mp4).
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        String savePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
            .endpointOverride(uri)
            .region(region)
            .build();

        getMediaObject(mediaStoreData, completePath, savePath);
        mediaStoreData.close();
    }
}
```

```
public static void getMediaObject(MediaStoreDataClient mediaStoreData, String
completePath, String savePath) {

    try {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .path(completePath)
            .build();

        // Write out the data to a file.
        ResponseInputStream<GetObjectResponse> data =
mediaStoreData.getObject(objectRequest);
        byte[] buffer = new byte[data.available()];
        data.read(buffer);

        File targetFile = new File(savePath);
        OutputStream outputStream = new FileOutputStream(targetFile);
        outputStream.write(buffer);
        System.out.println("The data was written to " + savePath);

    } catch (MediaStoreDataException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

private static String getEndpoint(String containerName) {
    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();

    DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
        .containerName(containerName)
        .build();

    DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
    return response.container().endpoint();
}
}
```



- API の詳細については、「API リファレンス [GetObject](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListContainers

次の例は、ListContainers を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.Container;
import software.amazon.awssdk.services.mediastore.model.ListContainersResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListContainers {

    public static void main(String[] args) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        listAllContainers(mediaStoreClient);
        mediaStoreClient.close();
    }
}
```

```
    }

    public static void listAllContainers(MediaStoreClient mediaStoreClient) {
        try {
            ListContainersResponse containersResponse =
mediaStoreClient.listContainers();
            List<Container> containers = containersResponse.containers();
            for (Container container : containers) {
                System.out.println("Container name is " + container.name());
            }

        } catch (MediaStoreException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [ListContainers](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## PutObject

次の例は、PutObject を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectResponse;
```

```
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import java.io.File;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObject {
    public static void main(String[] args) throws URISyntaxException {
        final String USAGE = ""

                To run this example, supply the name of a container, a file location
                to use, and path in the container\s

                Ex: <containerName> <filePath> <completePath>
                """;

        if (args.length < 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String containerName = args[0];
        String filePath = args[1];
        String completePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
            .endpointOverride(uri)
            .region(region)
            .build();

        putMediaObject(mediaStoreData, filePath, completePath);
        mediaStoreData.close();
    }
}
```

```
public static void putMediaObject(MediaStoreDataClient mediaStoreData, String
filePath, String completePath) {
    try {
        File myFile = new File(filePath);
        RequestBody requestBody = RequestBody.fromFile(myFile);

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .path(completePath)
            .contentType("video/mp4")
            .build();

        PutObjectResponse response = mediaStoreData.putObject(objectRequest,
requestBody);
        System.out.println("The saved object is " +
response.storageClass().toString());

    } catch (MediaStoreDataException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String getEndpoint(String containerName) {

    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();

    DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
        .containerName(containerName)
        .build();

    DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
    return response.container().endpoint();
}
}
```

- API の詳細については、「API リファレンス [PutObject](#)」の「」を参照してください。AWS SDK for Java 2.x

## OpenSearch SDK for Java 2.x を使用したサービス例

次のコード例は、AWS SDK for Java 2.x OpenSearch サービスで を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- [アクション](#)

### アクション

#### CreateDomain

次の例は、CreateDomain を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.VolumeType;
import software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.CreateDomainResponse;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDomain {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <domainName>

            Where:
                domainName - The name of the domain to create.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .build();

        createNewDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void createNewDomain(OpenSearchClient searchClient, String
domainName) {
        try {
            ClusterConfig clusterConfig = ClusterConfig.builder()
                .dedicatedMasterEnabled(true)
                .dedicatedMasterCount(3)
                .dedicatedMasterType("t2.small.search")
                .instanceType("t2.small.search")
                .instanceCount(5)
        }
    }
}
```

```
        .build();

        EBSOptions ebsOptions = EBSOptions.builder()
            .ebsEnabled(true)
            .volumeSize(10)
            .volumeType(VolumeType.GP2)
            .build();

        NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
            .enabled(true)
            .build();

        CreateDomainRequest domainRequest = CreateDomainRequest.builder()
            .domainName(domainName)
            .engineVersion("OpenSearch_1.0")
            .clusterConfig(clusterConfig)
            .ebsOptions(ebsOptions)
            .nodeToNodeEncryptionOptions(encryptionOptions)
            .build();

        System.out.println("Sending domain creation request...");
        CreateDomainResponse createResponse =
searchClient.createDomain(domainRequest);
        System.out.println("Domain status is " +
createResponse.domainStatus().toString());
        System.out.println("Domain Id is " +
createResponse.domainStatus().domainId());

        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- APIの詳細については、「APIリファレンス[CreateDomain](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteDomain

次の例は、DeleteDomainを使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDomain {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <domainName>

                Where:
                domainName - The name of the domain to delete.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
                .region(region)
                .build();
```



```
        deleteSpecificDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void deleteSpecificDomain(OpenSearchClient searchClient, String
domainName) {
        try {
            DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
                .domainName(domainName)
                .build();

            searchClient.deleteDomain(domainRequest);
            System.out.println(domainName + " was successfully deleted.");

        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [DeleteDomain](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListDomainNames

次の例は、ListDomainNames を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例  
を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
```

```
import software.amazon.awssdk.services.opensearch.model.DomainInfo;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesRequest;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesResponse;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListDomainNames {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();
        listAllDomains(searchClient);
        System.out.println("Done");
    }

    public static void listAllDomains(OpenSearchClient searchClient) {
        try {
            ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
                .engineType("OpenSearch")
                .build();

            ListDomainNamesResponse response =
searchClient.listDomainNames(namesRequest);
            List<DomainInfo> domainInfoList = response.domainNames();
            for (DomainInfo domain : domainInfoList)
                System.out.println("Domain name is " + domain.domainName());

        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [ListDomainNames](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## UpdateDomainConfig

次の例は、UpdateDomainConfig を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateDomain {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <domainName>

                Where:
                domainName - The name of the domain to update.
    }
```

```
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String domainName = args[0];
    Region region = Region.US_EAST_1;
    OpenSearchClient searchClient = OpenSearchClient.builder()
        .region(region)
        .build();

    updateSpecificDomain(searchClient, domainName);
    System.out.println("Done");
}

public static void updateSpecificDomain(OpenSearchClient searchClient, String
domainName) {
    try {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
            .build();

        UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();

        System.out.println("Sending domain update request...");
        UpdateDomainConfigResponse updateResponse =
searchClient.updateDomainConfig(updateDomainConfigRequest);
        System.out.println("Domain update response from Amazon OpenSearch
Service:");
        System.out.println(updateResponse.toString());

    } catch (OpenSearchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [UpdateDomainConfig](#)」の「」を参照してください。AWS SDK for Java 2.x

## EventBridge SDK for Java 2.x を使用した の例

次のコード例は、AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています EventBridge。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### 開始方法

こんにちは EventBridgeは

次のコード例は、の使用を開始する方法を示しています EventBridge。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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);
        }
    }
}
```

- API の詳細については、「API リファレンス [ListEventBuses](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### DeleteRule

次の例は、DeleteRule を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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");
}
```

- API の詳細については、「API リファレンス [DeleteRule](#)」の「」を参照してください。 AWS SDK for Java 2.x

### DescribeRule

次の例は、DescribeRule を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [DescribeRule](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DisableRule

次の例は、DisableRule を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例  
を見つけて、設定と実行の方法を確認してください。

そのルール名を使用してルールを無効化します。

```
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);
}
}
```

- API の詳細については、「API リファレンス [DisableRule](#)」の「」を参照してください。AWS SDK for Java 2.x

## EnableRule

次の例は、EnableRule を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

そのルール名を使用してルールを有効化します。

```
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);
}
}
```

- APIの詳細については、「APIリファレンス[EnableRule](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListRuleNamesByTarget

次の例は、ListRuleNamesByTarget を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

ターゲットを使用してすべてのルール名を一覧表示します。

```
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);
}
}
```

- API の詳細については、「API リファレンス [ListRuleNamesByTarget](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListRules

次の例は、ListRules を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

そのルール名を使用してルールを有効化します。

```
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);  
    }  
}
```

- API の詳細については、「API リファレンス [ListRules](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListTargetsByRule

次の例は、ListTargetsByRule を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

ルール名を使用してルールすべてのターゲットを一覧表示します。

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

- API の詳細については、「API リファレンス[ListTargetsByRule](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutEvents

次の例は、PutEvents を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
    String json = "{" +
        "\"UserEmail\": \"" + email + "\", " +
        "\"Message\": \"This event was generated by example code.\", " +
        "\"UtcTime\": \"Now.\" " +
        "}";

    PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
        .source("ExampleSource")
        .detail(json)
        .detailType("ExampleType")
        .build();

    PutEventsRequest eventsRequest = PutEventsRequest.builder()
        .entries(entry)
        .build();

    eventBrClient.putEvents(eventsRequest);
}
```

- API の詳細については、「API リファレンス[PutEvents](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutRule

次の例は、PutRule を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

スケジュールルールを作成します。

```
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);
    }
}
```

Amazon Simple Storage Service バケットにオブジェクトが追加されたときにトリガーするルールを作成します。

```
// Create a new event rule that triggers when an Amazon S3 object is created in
```

```
// a bucket.
public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
    String eventRuleName) {
    String pattern = "{\n" +
        "  \"source\": [\"aws.s3\"],\n" +
        "  \"detail-type\": [\"Object Created\"],\n" +
        "  \"detail\": {\n" +
        "    \"bucket\": {\n" +
        "      \"name\": [\"" + bucketName + "\"]}\n" +
        "    }\n" +
        "  }\n" +
        "}";

    try {
        PutRuleRequest ruleRequest = PutRuleRequest.builder()
            .description("Created by using the AWS SDK for Java v2")
            .name(eventRuleName)
            .eventPattern(pattern)
            .roleArn(roleArn)
            .build();

        PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
        System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());


    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [PutRule](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutTargets

次の例は、PutTargets を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

Amazon SNS トピックをルールターゲットとして追加します。

```
// 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 + ".");
}
```

ルールのターゲットにインプットトランスフォーマーを追加します。

```
public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
    String ruleName) {
    String targetId = java.util.UUID.randomUUID().toString();
```



```
InputTransformer inputTransformer = InputTransformer.builder()
    .inputTemplate("\Notification: sample event was received.\")
    .build();

Target target = Target.builder()
    .id(targetId)
    .arn(topicArn)
    .inputTransformer(inputTransformer)
    .build();

try {
    PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
        .rule(ruleName)
        .targets(target)
        .eventBusName(null)
        .build();

    eventBrClient.putTargets(targetsRequest);
} catch (EventBridgeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス[PutTargets](#)」の「」を参照してください。AWS SDK for Java 2.x

## RemoveTargets

次の例は、RemoveTargets を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

ルール名を使用してルールすべてのターゲットを削除します。

```
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");
    }
}
```

- API の詳細については、「API リファレンス [RemoveTargets](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## シナリオ

### ルールとターゲットの使用開始

次のコードサンプルは、以下の操作方法を示しています。

- ルールを作成して、ターゲットを追加する。
- ルールを有効化および無効化する。
- ルールとターゲットを一覧表示して更新する。
- イベントを送信して、リソースをクリーンアップする。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * This Java V2 example performs the following tasks with Amazon EventBridge:
 *
 * 1. Creates an AWS Identity and Access Management (IAM) role to use with
 * Amazon EventBridge.
 * 2. Amazon Simple Storage Service (Amazon S3) bucket with EventBridge events
 * enabled.
 * 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 * 4. Lists rules on the event bus.
 * 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and
 * lets the user subscribe to it.
 * 6. Adds a target to the rule that sends an email to the specified topic.
 * 7. Creates an EventBridge event that sends an email when an Amazon S3 object
 * is created.
 * 8. Lists Targets.
 * 9. Lists the rules for the same target.
 * 10. Triggers the rule by uploading a file to the Amazon S3 bucket.
 * 11. Disables a specific rule.
 * 12. Checks and print the state of the rule.
 * 13. Adds a transform to the rule to change the text of the email.
 * 14. Enables a specific rule.
 * 15. Triggers the updated rule by uploading a file to the Amazon S3 bucket.
 * 16. Updates the rule to be a custom rule pattern.
 * 17. Sending an event to trigger the rule.
 * 18. Cleans up resources.
```

```
*
*/
public class EventbridgeMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException, IOException
    {
        final String usage = ""

            Usage:
                <roleName> <bucketName> <topicName> <eventRuleName>

            Where:
                roleName - The name of the role to create.
                bucketName - The Amazon Simple Storage Service (Amazon S3)
bucket name to create.
                topicName - The name of the Amazon Simple Notification Service
(Amazon SNS) topic to create.
                eventRuleName - The Amazon EventBridge rule name to create.
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String polJSON = "{" +
            "\"Version\": \"2012-10-17\", " +
            "\"Statement\": [{" +
            "\"Effect\": \"Allow\", " +
            "\"Principal\": {" +
            "\"Service\": \"events.amazonaws.com\"" +
            "}, " +
            "\"Action\": \"sts:AssumeRole\"" +
            "}] " +
            "}";

        Scanner sc = new Scanner(System.in);
        String roleName = args[0];
        String bucketName = args[1];
        String topicName = args[2];
        String eventRuleName = args[3];

        Region region = Region.US_EAST_1;
```

```
EventBridgeClient eventBrClient = EventBridgeClient.builder()
    .region(region)
    .build();

S3Client s3Client = S3Client.builder()
    .region(region)
    .build();

Region regionGl = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(regionGl)
    .build();

SnsClient snsClient = SnsClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon EventBridge example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out
    .println("1. Create an AWS Identity and Access Management (IAM) role
to use with Amazon EventBridge.");
String roleArn = createIAMRole(iam, roleName, polJSON);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create an S3 bucket with EventBridge events
enabled.");
if (checkBucket(s3Client, bucketName)) {
    System.out.println("Bucket " + bucketName + " already exists. Ending
this scenario.");
    System.exit(1);
}

createBucket(s3Client, bucketName);
Thread.sleep(3000);
setBucketNotification(s3Client, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
System.out.println("3. Create a rule that triggers when an object is
uploaded to Amazon S3.");
Thread.sleep(10000);
addEventRule(eventBrClient, roleArn, bucketName, eventRuleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. List rules on the event bus.");
listRules(eventBrClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Create a new SNS topic for testing and let the user
subscribe to the topic.");
String topicArn = createSnsTopic(snsClient, topicName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Add a target to the rule that sends an email to the
specified topic.");
System.out.println("Enter your email to subscribe to the Amazon SNS
topic:");
String email = sc.nextLine();
subEmail(snsClient, topicArn, email);
System.out.println(
    "Use the link in the email you received to confirm your
subscription. Then, press Enter to continue.");
sc.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Create an EventBridge event that sends an email when
an Amazon S3 object is created.");
addSnsEventRule(eventBrClient, eventRuleName, topicArn, topicName,
eventRuleName, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 8. List Targets.");
listTargets(eventBrClient, eventRuleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 9. List the rules for the same target.");
```

```
listTargetRules(eventBrClient, topicArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 10. Trigger the rule by uploading a file to the S3
bucket.");
System.out.println("Press Enter to continue.");
sc.nextLine();
uploadTextFiletoS3(s3Client, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Disable a specific rule.");
changeRuleState(eventBrClient, eventRuleName, false);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Check and print the state of the rule.");
checkRule(eventBrClient, eventRuleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Add a transform to the rule to change the text of
the email.");
updateSnsEventRule(eventBrClient, topicArn, eventRuleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Enable a specific rule.");
changeRuleState(eventBrClient, eventRuleName, true);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 15. Trigger the updated rule by uploading a file to the
S3 bucket.");
System.out.println("Press Enter to continue.");
sc.nextLine();
uploadTextFiletoS3(s3Client, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 16. Update the rule to be a custom rule pattern.");
updateToCustomRule(eventBrClient, eventRuleName);
```

```
        System.out.println("Updated event rule " + eventRuleName + " to use a custom
pattern.");
        updateCustomRuleTargetWithTransform(eventBrClient, topicArn, eventRuleName);
        System.out.println("Updated event target " + topicArn + ".");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Sending an event to trigger the rule. This will
trigger a subscription email.");
        triggerCustomRule(eventBrClient, email);
        System.out.println("Events have been sent. Press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up resources.");
        System.out.println("Do you want to clean up resources (y/n)");
        String ans = sc.nextLine();
        if (ans.compareTo("y") == 0) {
            cleanupResources(eventBrClient, snsClient, s3Client, iam, topicArn,
eventRuleName, bucketName, roleName);
        } else {
            System.out.println("The resources will not be cleaned up. ");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon EventBridge example scenario has successfully
completed.");
        System.out.println(DASHES);
    }

    public static void cleanupResources(EventBridgeClient eventBrClient, SnsClient
snsClient, S3Client s3Client,
        IamClient iam, String topicArn, String eventRuleName, String bucketName,
String roleName) {
        System.out.println("Removing all targets from the event rule.");
        deleteTargetsFromRule(eventBrClient, eventRuleName);
        deleteRuleByName(eventBrClient, eventRuleName);
        deleteSNSTopic(snsClient, topicArn);
        deleteS3Bucket(s3Client, bucketName);
        deleteRole(iam, roleName);
    }
}
```



```
public static void deleteRole(IamClient iam, String roleName) {
    String policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess";
    DetachRolePolicyRequest policyRequest = DetachRolePolicyRequest.builder()
        .policyArn(policyArn)
        .roleName(roleName)
        .build();

    iam.detachRolePolicy(policyRequest);
    System.out.println("Successfully detached policy " + policyArn + " from role
" + roleName);

    // Delete the role.
    DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
        .roleName(roleName)
        .build();

    iam.deleteRole(roleRequest);
    System.out.println("*** Successfully deleted " + roleName);
}

public static void deleteS3Bucket(S3Client s3Client, String bucketName) {
    // Remove all the objects from the S3 bucket.
    ListObjectsRequest listObjects = ListObjectsRequest.builder()
        .bucket(bucketName)
        .build();

    ListObjectsResponse res = s3Client.listObjects(listObjects);
    List<S3Object> objects = res.contents();
    ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();

    for (S3Object myValue : objects) {
        toDelete.add(ObjectIdentifier.builder()
            .key(myValue.key())
            .build());
    }

    DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
        .bucket(bucketName)
        .delete(Delete.builder()
            .objects(toDelete).build())
        .build();

    s3Client.deleteObjects(dor);
}
```

```
// Delete the S3 bucket.
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucketName)
    .build();

s3Client.deleteBucket(deleteBucketRequest);
System.out.println("You have deleted the bucket and the objects");
}

// Delete the SNS topic.
public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
    try {
        DeleteTopicRequest request = DeleteTopicRequest.builder()
            .topicArn(topicArn)
            .build();

        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
    DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
        .name(ruleName)
        .build();

    eventBrClient.deleteRule(ruleRequest);
    System.out.println("Successfully deleted the rule");
}

public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    // First, get all targets that will be deleted.
    ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
        .rule(eventRuleName)
        .build();
```

```
ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
List<Target> allTargets = response.targets();

// Get all targets and delete them.
for (Target myTarget : allTargets) {
    RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
    .rule(eventRuleName)
    .ids(myTarget.id())
    .build();

    eventBrClient.removeTargets(removeTargetsRequest);
    System.out.println("Successfully removed the target");
}
}

public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
    String json = "{" +
        "\"UserEmail\": \"" + email + "\", " +
        "\"Message\": \"This event was generated by example code.\", " +
        "\"UtcTime\": \"Now.\" " +
        "}";

    PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
        .source("ExampleSource")
        .detail(json)
        .detailType("ExampleType")
        .build();

    PutEventsRequest eventsRequest = PutEventsRequest.builder()
        .entries(entry)
        .build();

    eventBrClient.putEvents(eventsRequest);
}

public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
    String ruleName) {
    String targetId = java.util.UUID.randomUUID().toString();
    InputTransformer inputTransformer = InputTransformer.builder()
        .inputTemplate("\"Notification: sample event was received.\"")
```

```
        .build();

    Target target = Target.builder()
        .id(targetId)
        .arn(topicArn)
        .inputTransformer(inputTransformer)
        .build();

    try {
        PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
            .rule(ruleName)
            .targets(target)
            .eventBusName(null)
            .build();

        eventBrClient.putTargets(targetsRequest);
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateToCustomRule(EventBridgeClient eventBrClient, String
ruleName) {
    String customEventsPattern = "{" +
        "\"source\": [\"ExampleSource\"]," +
        "\"detail-type\": [\"ExampleType\"]" +
        "}";

    PutRuleRequest request = PutRuleRequest.builder()
        .name(ruleName)
        .description("Custom test rule")
        .eventPattern(customEventsPattern)
        .build();

    eventBrClient.putRule(request);
}

// Update an Amazon S3 object created rule with a transform on the target.
public static void updateSnsEventRule(EventBridgeClient eventBrClient, String
topicArn, String ruleName) {
    String targetId = java.util.UUID.randomUUID().toString();
    Map<String, String> myMap = new HashMap<>();
    myMap.put("bucket", "$.detail.bucket.name");
```

```
myMap.put("time", "$.time");

InputTransformer inputTransformer = InputTransformer.builder()
    .inputTemplate("\Notification: an object was uploaded to bucket
<bucket> at <time>.\")
    .inputPathsMap(myMap)
    .build();

Target target = Target.builder()
    .id(targetId)
    .arn(topicArn)
    .inputTransformer(inputTransformer)
    .build();

try {
    PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
        .rule(ruleName)
        .targets(target)
        .eventBusName(null)
        .build();

    eventBrClient.putTargets(targetsRequest);

} catch (EventBridgeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

}

public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    try {
        DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
            .name(eventRuleName)
            .build();

        DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
        System.out.println("The state of the rule is " +
response.stateAsString());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }

    public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
        try {
            if (!isEnabled) {
                System.out.println("Disabling the rule: " + eventRuleName);
                DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                    .name(eventRuleName)
                    .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                    .name(eventRuleName)
                    .build();
                eventBrClient.enableRule(ruleRequest);
            }
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Create and upload a file to an S3 bucket to trigger an event.
    public static void uploadTextFiletoS3(S3Client s3Client, String bucketName)
throws IOException {
        // Create a unique file name.
        String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new
Date());
        String fileName = "TextFile" + fileSuffix + ".txt";

        File myFile = new File(fileName);
        FileWriter fw = new FileWriter(myFile.getAbsolutePath());
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write("This is a sample file for testing uploads.");
        bw.close();

        try {
            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(fileName)
```

```
        .build();

        s3Client.putObject(putOb, RequestBody.fromFile(myFile));

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
    ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
        .targetArn(topicArn)
        .build();

    ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
    List<String> rules = response.ruleNames();
    for (String rule : rules) {
        System.out.println("The rule name is " + rule);
    }
}

public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
{
    ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
        .rule(ruleName)
        .build();

    ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
    List<Target> targetsList = res.targets();
    for (Target target: targetsList) {
        System.out.println("Target ARN: "+target.arn());
    }
}

// Add a rule which triggers an SNS target when a file is uploaded to an S3
// bucket.
public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
    String topicName, String eventRuleName, String bucketName) {
```

```
String targetID = java.util.UUID.randomUUID().toString();
Target myTarget = Target.builder()
    .id(targetID)
    .arn(topicArn)
    .build();

List<Target> targets = new ArrayList<>();
targets.add(myTarget);
PutTargetsRequest request = PutTargetsRequest.builder()
    .eventBusName(null)
    .targets(targets)
    .rule(ruleName)
    .build();

eventBrClient.putTargets(request);
System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
target " + topicName + " for bucket "
    + bucketName + ".");
}

public static void subEmail(SnsClient snsClient, String topicArn, String email)
{
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("email")
            .endpoint(email)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
            + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void listRules(EventBridgeClient eventBrClient) {
    try {
        ListRulesRequest rulesRequest = ListRulesRequest.builder()
```



```

        .eventBusName("default")
        .limit(10)
        .build();

ListRulesResponse response = eventBrClient.listRules(rulesRequest);
List<Rule> rules = response.rules();
for (Rule rule : rules) {
    System.out.println("The rule name is : " + rule.name());
    System.out.println("The rule description is : " +
rule.description());
    System.out.println("The rule state is : " + rule.stateAsString());
}

} catch (EventBridgeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static String createSnsTopic(SnsClient snsClient, String topicName) {
    String topicPolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Sid\": \"EventBridgePublishTopic\"," +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": \"events.amazonaws.com\"" +
        "}," +
        "\"Resource\": \"*\"," +
        "\"Action\": \"sns:Publish\"" +
        "}]}" +
        "}";

    Map<String, String> topicAttributes = new HashMap<>();
    topicAttributes.put("Policy", topicPolicy);
    CreateTopicRequest topicRequest = CreateTopicRequest.builder()
        .name(topicName)
        .attributes(topicAttributes)
        .build();

    CreateTopicResponse response = snsClient.createTopic(topicRequest);
    System.out.println("Added topic " + topicName + " for email
subscriptions.");
    return response.topicArn();
}

```

```
}

// Create a new event rule that triggers when an Amazon S3 object is created in
// a bucket.
public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
    String eventRuleName) {
    String pattern = "{\n" +
        "  \"source\": [\"aws.s3\"],\n" +
        "  \"detail-type\": [\"Object Created\"],\n" +
        "  \"detail\": {\n" +
        "    \"bucket\": {\n" +
        "      \"name\": [\"" + bucketName + "\"]\n" +
        "    }\n" +
        "  }\n" +
        "}";

    try {
        PutRuleRequest ruleRequest = PutRuleRequest.builder()
            .description("Created by using the AWS SDK for Java v2")
            .name(eventRuleName)
            .eventPattern(pattern)
            .roleArn(roleArn)
            .build();

        PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
        System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Determine if the S3 bucket exists.
public static Boolean checkBucket(S3Client s3Client, String bucketName) {
    try {
        HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.headBucket(headBucketRequest);
        return true;
    }
```

```
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    return false;
}

// Set the S3 bucket notification configuration.
public static void setBucketNotification(S3Client s3Client, String bucketName) {
    try {
        EventBridgeConfiguration eventBridgeConfiguration =
EventBridgeConfiguration.builder()
            .build();

        NotificationConfiguration configuration =
NotificationConfiguration.builder()
            .eventBridgeConfiguration(eventBridgeConfiguration)
            .build();

        PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
            .builder()
            .bucket(bucketName)
            .notificationConfiguration(configuration)
            .skipDestinationValidation(true)
            .build();

        s3Client.putBucketNotificationConfiguration(configurationRequest);
        System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
    }
}
```

```
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(polJSON)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        AttachRolePolicyRequest rolePolicyRequest =
AttachRolePolicyRequest.builder()
            .roleName(rolename)
            .policyArn("arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess")
            .build();

        iam.attachRolePolicy(rolePolicyRequest);
        return response.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [DeleteRule](#)
  - [DescribeRule](#)
  - [DisableRule](#)
  - [EnableRule](#)
  - [ListRuleNamesByTarget](#)
  - [ListRules](#)
  - [ListTargetsByRule](#)
  - [PutEvents](#)
  - [PutRule](#)
  - [PutTargets](#)

## SDK for Java 2.x を使用した Forecast の例

次のコード例は、Forecast AWS SDK for Java 2.x でを使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- [アクション](#)

## アクション

### CreateDataset

次の例は、CreateDataset を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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;
}
}
```

- API の詳細については、「API リファレンス [CreateDataset](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateForecast

次の例は、CreateForecast を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。



```
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 "";
}
}
```

- API の詳細については、「API リファレンス [CreateForecast](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteDataset

次の例は、DeleteDataset を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [DeleteDataset](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteForecast

次の例は、DeleteForecast を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [DeleteForecast](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DescribeForecast

次の例は、DescribeForecast を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [DescribeForecast](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListDatasetGroups

次の例は、ListDatasetGroups を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [ListDatasetGroups](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListForecasts

次の例は、ListForecasts を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- APIの詳細については、「APIリファレンス[ListForecasts](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## AWS Glue SDK for Java 2.x を使用した の例

次のコード例は、AWS SDK for Java 2.x を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS Glue。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### 開始方法

こんにちは AWS Glueは

次のコード例は、AWS Glueの使用を開始する方法を示しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
package com.example.glue;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.ListJobsRequest;
import software.amazon.awssdk.services.glue.model.ListJobsResponse;
import java.util.List;

public class HelloGlue {
    public static void main(String[] args) {
        GlueClient glueClient = GlueClient.builder()
            .region(Region.US_EAST_1)
            .build();
```

```
        listJobs(glueClient);
    }

    public static void listJobs(GlueClient glueClient) {
        ListJobsRequest request = ListJobsRequest.builder()
            .maxResults(10)
            .build();
        ListJobsResponse response = glueClient.listJobs(request);
        List<String> jobList = response.jobNames();
        jobList.forEach(job -> {
            System.out.println("Job Name: " + job);
        });
    }
}
```

- APIの詳細については、「APIリファレンス[ListJobs](#)」の「」を参照してください。AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### CreateCrawler

次の例は、CreateCrawler を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
```

```
import software.amazon.awssdk.services.glue.model.CreateCrawlerRequest;
import software.amazon.awssdk.services.glue.model.CrawlerTargets;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.S3Target;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCrawler {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <IAM> <s3Path> <cron> <dbName> <crawlerName>

            Where:
                IAM - The ARN of the IAM role that has AWS Glue and S3
permissions.\s
                s3Path - The Amazon Simple Storage Service (Amazon S3) target
that contains data (for example, CSV data).
                cron - A cron expression used to specify the schedule (i.e.,
cron(15 12 * * ? *).
                dbName - The database name.\s
                crawlerName - The name of the crawler.\s
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        Region region = Region.US_EAST_1;
```

```
GlueClient glueClient = GlueClient.builder()
    .region(region)
    .build();

createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
glueClient.close();
}

public static void createGlueCrawler(GlueClient glueClient,
    String iam,
    String s3Path,
    String cron,
    String dbName,
    String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)
            .build();

        // Add the S3Target to a list.
        List<S3Target> targetList = new ArrayList<>();
        targetList.add(s3Target);

        CrawlerTargets targets = CrawlerTargets.builder()
            .s3Targets(targetList)
            .build();

        CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
            .databaseName(dbName)
            .name(crawlerName)
            .description("Created by the AWS Glue Java API")
            .targets(targets)
            .role(iam)
            .schedule(cron)
            .build();

        glueClient.createCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully created");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}  
}
```

- API の詳細については、「API リファレンス [CreateCrawler](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## GetCrawler

次の例は、GetCrawler を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.glue.GlueClient;  
import software.amazon.awssdk.services.glue.model.GetCrawlerRequest;  
import software.amazon.awssdk.services.glue.model.GetCrawlerResponse;  
import software.amazon.awssdk.services.glue.model.GlueException;  
import java.time.Instant;  
import java.time.ZoneId;  
import java.time.format.DateTimeFormatter;  
import java.time.format.FormatStyle;  
import java.util.Locale;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class GetCrawler {  
    public static void main(String[] args) {  
        final String usage = ""
```

```
Usage:
    <crawlerName>

Where:
    crawlerName - The name of the crawler.\s
    """";

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String crawlerName = args[0];
Region region = Region.US_EAST_1;
GlueClient glueClient = GlueClient.builder()
    .region(region)
    .build();

getSpecificCrawler(glueClient, crawlerName);
glueClient.close();
}

public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
{
    try {
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
        Instant createDate = response.crawler().creationTime();

        // Convert the Instant to readable date
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the Crawler is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```



```
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [GetCrawler](#)」の「」を参照してください。AWS SDK for Java 2.x

## GetDatabase

次の例は、GetDatabase を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetDatabaseRequest;
import software.amazon.awssdk.services.glue.model.GetDatabaseResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetDatabase {
```

```
public static void main(String[] args) {
    final String usage = ""

        Usage:
            <databaseName>

        Where:
            databaseName - The name of the database.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String databaseName = args[0];
    Region region = Region.US_EAST_1;
    GlueClient glueClient = GlueClient.builder()
        .region(region)
        .build();

    getSpecificDatabase(glueClient, databaseName);
    glueClient.close();
}

public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
    try {
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
            .name(databaseName)
            .build();

        GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
        Instant createDate = response.database().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the database is " + createDate);
    }
}
```

```
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [GetDatabase](#)」の「」を参照してください。AWS SDK for Java 2.x

## GetTables

次の例は、GetTables を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetTableRequest;
import software.amazon.awssdk.services.glue.model.GetTableResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
*/
public class GetTable {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbName> <tableName>

            Where:
                dbName - The database name.\s
                tableName - The name of the table.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbName = args[0];
        String tableName = args[1];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
            .region(region)
            .build();

        getGlueTable(glueClient, dbName, tableName);
        glueClient.close();
    }

    public static void getGlueTable(GlueClient glueClient, String dbName, String
tableName) {
        try {
            GetTableRequest tableRequest = GetTableRequest.builder()
                .databaseName(dbName)
                .name(tableName)
                .build();

            GetTableResponse tableResponse = glueClient.getTable(tableRequest);
            Instant createDate = tableResponse.table().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
                DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)

```

```
        .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the table is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [GetTables](#)」の「」を参照してください。AWS SDK for Java 2.x

## StartCrawler

次の例は、StartCrawler を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.StartCrawlerRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class StartCrawler {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <crawlerName>

            Where:
                crawlerName - The name of the crawler.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String crawlerName = args[0];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
            .region(region)
            .build();

        startSpecificCrawler(glueClient, crawlerName);
        glueClient.close();
    }

    public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.startCrawler(crawlerRequest);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [StartCrawler](#)」の「」を参照してください。AWS SDK for Java 2.x

## シナリオ

### クローラーとジョブを開始する

次のコードサンプルは、以下の操作方法を示しています。

- パブリック Amazon S3 バケットをクロールし、CSV 形式のメタデータのデータベースを生成するクローラーを作成する。
- のデータベースとテーブルに関する情報を一覧表示します AWS Glue Data Catalog。
- S3 バケットから CSV 形式のデータを抽出するジョブを作成し、そのデータを変換して JSON 形式の出力を別の S3 バケットにロードする。
- ジョブ実行に関する情報を一覧表示し、変換されたデータを表示してリソースをクリーンアップする。

詳細については、「[チュートリアル: AWS Glue Studio の開始方法](#)」を参照してください。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To set up the resources, see this documentation topic:
 *
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
```

```

*
* This example performs the following tasks:
*
* 1. Create a database.
* 2. Create a crawler.
* 3. Get a crawler.
* 4. Start a crawler.
* 5. Get a database.
* 6. Get tables.
* 7. Create a job.
* 8. Start a job run.
* 9. List all jobs.
* 10. Get job runs.
* 11. Delete a job.
* 12. Delete a database.
* 13. Delete a crawler.
*/

public class GlueScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>\s

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

                """;

        if (args.length != 9) {

```



```
        System.out.println(usage);
        System.exit(1);
    }

    String iam = args[0];
    String s3Path = args[1];
    String cron = args[2];
    String dbName = args[3];
    String crawlerName = args[4];
    String jobName = args[5];
    String scriptLocation = args[6];
    String locationUri = args[7];
    String bucketNameSc = args[8];

    Region region = Region.US_EAST_1;
    GlueClient glueClient = GlueClient.builder()
        .region(region)
        .build();
    System.out.println(DASHES);
    System.out.println("Welcome to the AWS Glue scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Create a database.");
    createDatabase(glueClient, dbName, locationUri);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Create a crawler.");
    createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Get a crawler.");
    getSpecificCrawler(glueClient, crawlerName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Start a crawler.");
    startSpecificCrawler(glueClient, crawlerName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("5. Get a database.");
```

```
getSpecificDatabase(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("*** Wait 5 min for the tables to become available");
TimeUnit.MINUTES.sleep(5);
System.out.println("6. Get tables.");
String myTableName = getGlueTables(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Create a job.");
createJob(glueClient, jobName, iam, scriptLocation);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Start a Job run.");
startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. List all jobs.");
getAllJobs(glueClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get job runs.");
getJobRuns(glueClient, jobName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Delete a job.");
deleteJob(glueClient, jobName);
System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
TimeUnit.MINUTES.sleep(5);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Delete a database.");
deleteDatabase(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Delete a crawler.");
```

```
        deleteSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }

    public static void createDatabase(GlueClient glueClient, String dbName, String
locationUri) {
        try {
            DatabaseInput input = DatabaseInput.builder()
                .description("Built with the AWS SDK for Java V2")
                .name(dbName)
                .locationUri(locationUri)
                .build();

            CreateDatabaseRequest request = CreateDatabaseRequest.builder()
                .databaseInput(input)
                .build();

            glueClient.createDatabase(request);
            System.out.println(dbName + " was successfully created");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createGlueCrawler(GlueClient glueClient,
        String iam,
        String s3Path,
        String cron,
        String dbName,
        String crawlerName) {

        try {
            S3Target s3Target = S3Target.builder()
                .path(s3Path)
                .build();

            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);
```

```
CrawlerTargets targets = CrawlerTargets.builder()
    .s3Targets(targetList)
    .build();

CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
    .databaseName(dbName)
    .name(crawlerName)
    .description("Created by the AWS Glue Java API")
    .targets(targets)
    .role(iam)
    .schedule(cron)
    .build();

glueClient.createCrawler(crawlerRequest);
System.out.println(crawlerName + " was successfully created");

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
{
    try {
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        boolean ready = false;
        while (!ready) {
            GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
            String status = response.crawler().stateAsString();
            if (status.compareTo("READY") == 0) {
                ready = true;
            }
            Thread.sleep(3000);
        }

        System.out.println("The crawler is now ready");

    } catch (GlueException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
  
  public static void startSpecificCrawler(GlueClient glueClient, String  
crawlerName) {  
    try {  
      StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()  
        .name(crawlerName)  
        .build();  
  
      glueClient.startCrawler(crawlerRequest);  
      System.out.println(crawlerName + " was successfully started!");  
  
    } catch (GlueException e) {  
      System.err.println(e.awsErrorDetails().errorMessage());  
      System.exit(1);  
    }  
  }  
}  
  
public static void getSpecificDatabase(GlueClient glueClient, String  
databaseName) {  
  try {  
    GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()  
      .name(databaseName)  
      .build();  
  
    GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);  
    Instant createDate = response.database().createTime();  
  
    // Convert the Instant to readable date.  
    DateTimeFormatter formatter =  
    DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)  
      .withLocale(Locale.US)  
      .withZone(ZoneId.systemDefault());  
  
    formatter.format(createDate);  
    System.out.println("The create date of the database is " + createDate);  
  
  } catch (GlueException e) {  
    System.err.println(e.awsErrorDetails().errorMessage());  
    System.exit(1);  
  }  
}
```

```
public static String getGlueTables(GlueClient glueClient, String dbName) {
    String myTableName = "";
    try {
        GetTablesRequest tableRequest = GetTablesRequest.builder()
            .databaseName(dbName)
            .build();

        GetTablesResponse response = glueClient.getTables(tableRequest);
        List<Table> tables = response.tableList();
        if (tables.isEmpty()) {
            System.out.println("No tables were returned");
        } else {
            for (Table table : tables) {
                myTableName = table.name();
                System.out.println("Table name is: " + myTableName);
            }
        }
    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return myTableName;
}

public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
    String outBucket) {
    try {
        Map<String, String> myMap = new HashMap<>();
        myMap.put("--input_database", inputDatabase);
        myMap.put("--input_table", inputTable);
        myMap.put("--output_bucket_url", outBucket);

        StartJobRunRequest runRequest = StartJobRunRequest.builder()
            .workerType(WorkerType.G_1_X)
            .numberOfWorkers(10)
            .arguments(myMap)
            .jobName(jobName)
            .build();

        StartJobRunResponse response = glueClient.startJobRun(runRequest);
        System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());
    }
}
```

```
    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createJob(GlueClient glueClient, String jobName, String iam,
String scriptLocation) {
    try {
        JobCommand command = JobCommand.builder()
            .pythonVersion("3")
            .name("glueetl")
            .scriptLocation(scriptLocation)
            .build();

        CreateJobRequest jobRequest = CreateJobRequest.builder()
            .description("A Job created by using the AWS SDK for Java V2")
            .glueVersion("2.0")
            .workerType(WorkerType.G_1_X)
            .numberOfWorkers(10)
            .name(jobName)
            .role(iam)
            .command(command)
            .build();

        glueClient.createJob(jobRequest);
        System.out.println(jobName + " was successfully created.");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getAllJobs(GlueClient glueClient) {
    try {
        GetJobsRequest jobsRequest = GetJobsRequest.builder()
            .maxResults(10)
            .build();

        GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
        List<Job> jobs = jobsResponse.jobs();
        for (Job job : jobs) {
```

```
        System.out.println("Job name is : " + job.name());
        System.out.println("The job worker type is : " +
job.workerType().name());
    }

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getJobRuns(GlueClient glueClient, String jobName) {
    try {
        GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
            .jobName(jobName)
            .maxResults(20)
            .build();

        boolean jobDone = false;
        while (!jobDone) {
            GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
            List<JobRun> jobRuns = response.jobRuns();
            for (JobRun jobRun : jobRuns) {
                String jobState = jobRun.jobRunState().name();
                if (jobState.compareTo("SUCCEEDED") == 0) {
                    System.out.println(jobName + " has succeeded");
                    jobDone = true;

                } else if (jobState.compareTo("STOPPED") == 0) {
                    System.out.println("Job run has stopped");
                    jobDone = true;

                } else if (jobState.compareTo("FAILED") == 0) {
                    System.out.println("Job run has failed");
                    jobDone = true;

                } else if (jobState.compareTo("TIMEOUT") == 0) {
                    System.out.println("Job run has timed out");
                    jobDone = true;

                } else {
                    System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                    System.out.println("Job run Id is " + jobRun.id());
                }
            }
        }
    }
}
```



```
        System.out.println("The Glue version is " +
jobRun.glueVersion());
    }
    TimeUnit.SECONDS.sleep(5);
}
}

} catch (GlueException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void deleteJob(GlueClient glueClient, String jobName) {
    try {
        DeleteJobRequest jobRequest = DeleteJobRequest.builder()
            .jobName(jobName)
            .build();

        glueClient.deleteJob(jobRequest);
        System.out.println(jobName + " was successfully deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteDatabase(GlueClient glueClient, String databaseName) {
    try {
        DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
            .name(databaseName)
            .build();

        glueClient.deleteDatabase(request);
        System.out.println(databaseName + " was successfully deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

```
public static void deleteSpecificCrawler(GlueClient glueClient, String
crawlerName) {
    try {
        DeleteCrawlerRequest deleteCrawlerRequest =
DeleteCrawlerRequest.builder()
                        .name(crawlerName)
                        .build();

        glueClient.deleteCrawler(deleteCrawlerRequest);
        System.out.println(crawlerName + " was deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [CreateCrawler](#)
  - [CreateJob](#)
  - [DeleteCrawler](#)
  - [DeleteDatabase](#)
  - [DeleteJob](#)
  - [DeleteTable](#)
  - [GetCrawler](#)
  - [GetDatabase](#)
  - [GetDatabases](#)
  - [GetJob](#)
  - [GetJobRun](#)
  - [GetJobRuns](#)
  - [GetTables](#)
  - [ListJobs](#)
  - [StartCrawler](#)

- [StartJobRun](#)

## HealthImaging SDK for Java 2.x を使用した の例

次のコード例は、AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています HealthImaging。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### CopyImageSet

次の例は、CopyImageSet を使用する方法を説明しています。

#### SDK for Java 2.x

```
public static String copyMedicalImageSet(MedicalImagingClient
medicalImagingClient,
    String datastoreId,
    String imageSetId,
    String latestVersionId,
    String destinationImageSetId,
    String destinationVersionId) {

    try {
        CopySourceImageSetInformation copySourceImageSetInformation =
CopySourceImageSetInformation.builder()
```

```
        .latestVersionId(latestVersionId)
        .build();

    CopyImageSetInformation.Builder copyImageSetBuilder =
CopyImageSetInformation.builder()
        .sourceImageSet(copySourceImageSetInformation);

    if (destinationImageSetId != null) {
        copyImageSetBuilder =
copyImageSetBuilder.destinationImageSet(CopyDestinationImageSet.builder()
            .imageSetId(destinationImageSetId)
            .latestVersionId(destinationVersionId)
            .build());
    }

    CopyImageSetRequest copyImageSetRequest = CopyImageSetRequest.builder()
        .datastoreId(datastoreId)
        .sourceImageSetId(imageSetId)
        .copyImageSetInformation(copyImageSetBuilder.build())
        .build();

    CopyImageSetResponse response =
medicalImagingClient.copyImageSet(copyImageSetRequest);

    return response.destinationImageSetProperties().imageSetId();
} catch (MedicalImagingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return "";
}
```

- API の詳細については、「API リファレンス [CopyImageSet](#)」の「」を参照してください。  
AWS SDK for Java 2.x

#### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## CreateDatastore

次の例は、CreateDatastore を使用する方法を説明しています。

SDK for Java 2.x

```
public static String createMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
    String datastoreName) {
    try {
        CreateDatastoreRequest datastoreRequest =
CreateDatastoreRequest.builder()
            .datastoreName(datastoreName)
            .build();
        CreateDatastoreResponse response =
medicalImagingClient.createDatastore(datastoreRequest);
        return response.datastoreId();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

- API の詳細については、「API リファレンス [CreateDatastore](#)」の「」を参照してください。  
AWS SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## DeleteDatastore

次の例は、DeleteDatastore を使用する方法を説明しています。

## SDK for Java 2.x

```
public static void deleteMedicalImagingDatastore(MedicalImagingClient
medicalImagingClient,
    String datastoreID) {
    try {
        DeleteDatastoreRequest datastoreRequest =
DeleteDatastoreRequest.builder()
        .datastoreId(datastoreID)
        .build();
        medicalImagingClient.deleteDatastore(datastoreRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DeleteDatastore](#)」の「」を参照してください。  
AWS SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例  
を見つけて、設定と実行の方法を確認してください。

**DeleteImageSet**

次の例は、DeleteImageSet を使用する方法を説明しています。

## SDK for Java 2.x

```
public static void deleteMedicalImageSet(MedicalImagingClient
medicalImagingClient,
    String datastoreId,
    String imagesetId) {
    try {
        DeleteImageSetRequest deleteImageSetRequest =
DeleteImageSetRequest.builder()
        .datastoreId(datastoreId)
        .imageSetId(imagesetId)
```

```
        .build();

        medicalImagingClient.deleteImageSet(deleteImageSetRequest);

        System.out.println("The image set was deleted.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DeleteImageSet](#)」の「」を参照してください。  
AWS SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## GetDICOMImportJob

次の例は、GetDICOMImportJob を使用する方法を説明しています。

SDK for Java 2.x

```
public static DICOMImportJobProperties getDicomImportJob(MedicalImagingClient
medicalImagingClient,
                String datastoreId,
                String jobId) {

    try {
        GetDicomImportJobRequest getDicomImportJobRequest =
        GetDicomImportJobRequest.builder()
            .datastoreId(datastoreId)
            .jobId(jobId)
            .build();

        GetDicomImportJobResponse response =
        medicalImagingClient.getDICOMImportJob(getDicomImportJobRequest);
        return response.jobProperties();
    } catch (MedicalImagingException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- API の詳細については、「API リファレンス [GetDICOMImportJob](#)」を参照してください。  
AWS SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## GetDatastore

次の例は、GetDatastore を使用する方法を説明しています。


SDK for Java 2.x

```
public static DatastoreProperties getMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
    String datastoreID) {
    try {
        GetDatastoreRequest datastoreRequest = GetDatastoreRequest.builder()
            .datastoreId(datastoreID)
            .build();
        GetDatastoreResponse response =
medicalImagingClient.getDatastore(datastoreRequest);
        return response.datastoreProperties();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```



- API の詳細については、「API リファレンス [GetDatastore](#)」の「」を参照してください。  
AWS SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## GetImageFrame

次の例は、GetImageFrame を使用する方法を説明しています。

SDK for Java 2.x

```
public static void getMedicalImageSetFrame(MedicalImagingClient
medicalImagingClient,
    String destinationPath,
    String datastoreId,
    String imagesetId,
    String imageFrameId) {

    try {
        GetImageFrameRequest getImageSetMetadataRequest =
        GetImageFrameRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId)
            .imageFrameInformation(ImageFrameInformation.builder()
                .imageFrameId(imageFrameId)
                .build())
            .build();

        medicalImagingClient.getImageFrame(getImageSetMetadataRequest,
        FileSystems.getDefault().getPath(destinationPath));

        System.out.println("Image frame downloaded to " +
        destinationPath);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
}
```

- API の詳細については、「API リファレンス [GetImageFrame](#)」の「」を参照してください。  
AWS SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## GetImageSet

次の例は、GetImageSet を使用する方法を説明しています。

SDK for Java 2.x

```
public static GetImageSetResponse getMedicalImageSet(MedicalImagingClient  
medicalImagingClient,  
    String datastoreId,  
    String imagesetId,  
    String versionId) {  
    try {  
        GetImageSetRequest.Builder getImageSetRequestBuilder =  
        GetImageSetRequest.builder()  
            .datastoreId(datastoreId)  
            .imageSetId(imagesetId);  
  
        if (versionId != null) {  
            getImageSetRequestBuilder =  
            getImageSetRequestBuilder.versionId(versionId);  
        }  
  
        return  
        medicalImagingClient.getImageSet(getImageSetRequestBuilder.build());  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

```
        return null;
    }
```

- API の詳細については、「API リファレンス [GetImageSet](#)」の「」を参照してください。AWS SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## GetImageSetMetadata

次の例は、GetImageSetMetadata を使用する方法を説明しています。

SDK for Java 2.x

```
public static void getMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
        String destinationPath,
        String datastoreId,
        String imagesetId,
        String versionId) {

    try {
        GetImageSetMetadataRequest.Builder getImageSetMetadataRequestBuilder =
        GetImageSetMetadataRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId);

        if (versionId != null) {
            getImageSetMetadataRequestBuilder =
            getImageSetMetadataRequestBuilder.versionId(versionId);
        }

        medicalImagingClient.getImageSetMetadata(getImageSetMetadataRequestBuilder.build(),
            FileSystems.getDefault().getPath(destinationPath));

        System.out.println("Metadata downloaded to " + destinationPath);
    }
}
```

```
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [GetImageSetMetadata](#)」の「」を参照してください。AWS SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## ListDICOMImportJobs

次の例は、ListDICOMImportJobs を使用する方法を説明しています。


SDK for Java 2.x

```
public static List<DICOMImportJobSummary>
listDicomImportJobs(MedicalImagingClient medicalImagingClient,
    String datastoreId) {

    try {
        ListDicomImportJobsRequest listDicomImportJobsRequest =
ListDicomImportJobsRequest.builder()
            .datastoreId(datastoreId)
            .build();
        ListDicomImportJobsResponse response =
medicalImagingClient.listDICOMImportJobs(listDicomImportJobsRequest);
        return response.jobSummaries();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return new ArrayList<>();
}
```

- API の詳細については、「API リファレンス [ListDICOMImportJobs](#)」を参照してください。  
AWS SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## ListDatastores

次の例は、ListDatastores を使用する方法を説明しています。

SDK for Java 2.x

```
public static List<DatastoreSummary>
listMedicalImagingDatastores(MedicalImagingClient medicalImagingClient) {
    try {
        ListDatastoresRequest datastoreRequest = ListDatastoresRequest.builder()
            .build();
        ListDatastoresIterable responses =
medicalImagingClient.listDatastoresPaginator(datastoreRequest);
        List<DatastoreSummary> datastoreSummaries = new ArrayList<>();

        responses.stream().forEach(response ->
datastoreSummaries.addAll(response.datastoreSummaries()));

        return datastoreSummaries;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- API の詳細については、「API リファレンス [ListDatastores](#)」の「」を参照してください。  
AWS SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## ListImageSetVersions

次の例は、ListImageSetVersions を使用する方法を説明しています。

SDK for Java 2.x

```
public static List<ImageSetProperties>
listMedicalImageSetVersions(MedicalImagingClient medicalImagingClient,
    String datastoreId,
    String imagesetId) {
    try {
        ListImageSetVersionsRequest getImageSetRequest =
ListImageSetVersionsRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId)
            .build();

        ListImageSetVersionsIterable responses = medicalImagingClient
            .listImageSetVersionsPaginator(getImageSetRequest);
        List<ImageSetProperties> imageSetProperties = new ArrayList<>();
        responses.stream().forEach(response ->
imageSetProperties.addAll(response.imageSetPropertiesList()));

        return imageSetProperties;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- API の詳細については、「API リファレンス [ListImageSetVersions](#)」の「」を参照してください。 AWS SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## ListTagsForResource

次の例は、ListTagsForResource を使用する方法を説明しています。

SDK for Java 2.x

```
public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
    String resourceArn) {
    try {
        ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
            .resourceArn(resourceArn)
            .build();

        return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- API の詳細については、「API リファレンス [ListTagsForResource](#)」の「」を参照してください。 AWS SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## SearchImageSets

次の例は、SearchImageSets を使用する方法を説明しています。

SDK for Java 2.x

イメージセットを検索するためのユーティリティ関数。

```
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;
}
```

ユースケース #1: EQUAL 演算子。

```
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();
    }
}

```

## ユースケース #2: DICOM StudyDate と DICOM を使用する BETWEEN 演算子StudyTime。

```

    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();
}
```

ユースケース #3: `createdAt` を使用する BETWEEN 演算子。タイムスタディは以前に永続化されています。

```
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();
}
```

ユースケース #4: DICOM SeriesInstanceUID の EQUAL 演算子と `updatedAt` の BETWEEN、および `updatedAt` フィールドの ASC 順でのソートレスポンス。

```
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();
}
```

- APIの詳細については、「APIリファレンス[SearchImageSets](#)」の「」を参照してください。  
AWS SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

**StartDICOMImportJob**

次の例は、StartDICOMImportJob を使用する方法を説明しています。

SDK for Java 2.x

```
public static String startDicomImportJob(MedicalImagingClient
medicalImagingClient,
    String jobName,
    String datastoreId,
    String dataAccessRoleArn,
    String inputS3Uri,
    String outputS3Uri) {

    try {
        StartDicomImportJobRequest startDicomImportJobRequest =
StartDicomImportJobRequest.builder()
            .jobName(jobName)
            .datastoreId(datastoreId)
            .dataAccessRoleArn(dataAccessRoleArn)
            .inputS3Uri(inputS3Uri)
            .outputS3Uri(outputS3Uri)
            .build();
        StartDicomImportJobResponse response =
medicalImagingClient.startDICOMImportJob(startDicomImportJobRequest);
        return response.jobId();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

- API の詳細については、「API リファレンス」の[StartDICOMImportJob](#)を参照してください。AWS SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## TagResource

次の例は、TagResource を使用する方法を説明しています。

SDK for Java 2.x

```
public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
    String resourceArn,
    Map<String, String> tags) {
    try {
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tags(tags)
            .build();

        medicalImagingClient.tagResource(tagResourceRequest);

        System.out.println("Tags have been added to the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス[TagResource](#)」の「」を参照してください。AWS SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## UntagResource

次の例は、UntagResource を使用する方法を説明しています。

### SDK for Java 2.x

```
public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
        String resourceArn,
        Collection<String> tagKeys) {
    try {
        UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tagKeys(tagKeys)
            .build();

        medicalImagingClient.untagResource(untagResourceRequest);

        System.out.println("Tags have been removed from the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [UntagResource](#)」の「」を参照してください。  
AWS SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## UpdateImageSetMetadata

次の例は、UpdateImageSetMetadata を使用する方法を説明しています。

SDK for Java 2.x

```
public static void updateMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
                                                String datastoreId,
                                                String imagesetId,
                                                String versionId,
                                                MetadataUpdates
metadataUpdates) {
    try {
        UpdateImageSetMetadataRequest updateImageSetMetadataRequest =
UpdateImageSetMetadataRequest
            .builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId)
            .latestVersionId(versionId)
            .updateImageSetMetadataUpdates(metadataUpdates)
            .build();

        UpdateImageSetMetadataResponse response =
medicalImagingClient.updateImageSetMetadata(updateImageSetMetadataRequest);

        System.out.println("The image set metadata was updated" + response);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

ユースケース #1: 属性を挿入または更新します。

```
final String insertAttributes = ""
{
    "SchemaVersion": 1.1,
    "Study": {
        "DICOM": {
            "StudyDescription": "CT CHEST"
        }
    }
}
```

```
        }
    }
    """;

    MetadataUpdates metadataInsertUpdates = MetadataUpdates.builder()
        .dicomUpdates(DICOMUpdates.builder()
            .updateableAttributes(SdkBytes.fromByteBuffer(
                ByteBuffer.wrap(insertAttributes
                    .getBytes(StandardCharsets.UTF_8))))
            .build())
        .build();

    updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
        imagesetId,
        versionid, metadataInsertUpdates);
```

ユースケース #2: 属性を削除します。

```
    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);
```

ユースケース #3: インスタンスを削除します。



```

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);

```

- APIの詳細については、「API リファレンス [UpdateImageSetMetadata](#)」の「」を参照してください。AWS SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## シナリオ

### データストアにタグを付ける

次のコード例は、HealthImaging データストアにタグを付ける方法を示しています。

## SDK for Java 2.x

データストアにタグを付けます。

```
final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

TagResource.tagMedicalImagingResource(medicalImagingClient,
datastoreArn,
                                     ImmutableMap.of("Deployment", "Development"));
```

リソースにタグを付けるためのユーティリティ関数。

```
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);
    }
}
```

データストアのタグを一覧表示します。

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

リソースのタグを一覧表示するユーティリティ関数。

```
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;
}
```

データストアのタグを解除するには

```
final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

UntagResource.untagMedicalImagingResource(medicalImagingClient,
    datastoreArn,
        Collections.singletonList("Deployment"));
```

リソースのタグを解除するユーティリティ関数。

```
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);
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [ListTagsForResource](#)
  - [TagResource](#)
  - [UntagResource](#)

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## イメージセットにタグを付ける

次のコード例は、HealthImaging 画像セットにタグを付ける方法を示しています。

### SDK for Java 2.x

イメージセットにタグを付けるには

```
        final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
```

```
TagResource.tagMedicalImagingResource(medicalImagingClient,  
imageSetArn,  
ImmutableMap.of("Deployment", "Development"));
```

リソースにタグを付けるためのユーティリティ関数。

```
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);  
    }  
}
```

イメージセットのタグを一覧表示します。

```
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());  
}
```

リソースのタグを一覧表示するユーティリティ関数。

```
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;
}
```

イメージセットのタグを解除します。

```
final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

UntagResource.untagMedicalImagingResource(medicalImagingClient,
    imageSetArn,
        Collections.singletonList("Deployment"));
```

リソースのタグを解除するユーティリティ関数。

```
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);
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [ListTagsForResource](#)
  - [TagResource](#)
  - [UntagResource](#)

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## SDK for Java 2.x を使用した IAM の例

次のコード例は、IAM AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

## 開始方法

### IAM へようこそ

次のコード例は、IAM の使用を開始する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.ListPoliciesResponse;
import software.amazon.awssdk.services.iam.model.Policy;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloIAM {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listPolicies(iam);
    }

    public static void listPolicies(IamClient iam) {
        ListPoliciesResponse response = iam.listPolicies();
        List<Policy> polList = response.policies();
        polList.forEach(policy -> {
            System.out.println("Policy Name: " + policy.policyName());
        });
    }
}
```



```
    });  
  }  
}
```

- API の詳細については、「API リファレンス [ListPolicies](#)」の「」を参照してください。AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### AttachRolePolicy

次の例は、AttachRolePolicy を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.iam.IamClient;  
import software.amazon.awssdk.services.iam.model.IamException;  
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;  
import software.amazon.awssdk.services.iam.model.AttachedPolicy;  
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;  
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;  
import java.util.List;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic: */
```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class AttachRolePolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <roleName> <policyArn>\s

            Where:
                roleName - A role name that you can obtain from the AWS
Management Console.\s
                policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        attachIAMRolePolicy(iam, roleName, policyArn);
        iam.close();
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                .roleName(roleName)
                .build();

            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        }
    }
}
```

```
List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

// Ensure that the policy is not attached to this role
String polArn = "";
for (AttachedPolicy policy : attachedPolicies) {
    polArn = policy.policyArn();
    if (polArn.compareTo(policyArn) == 0) {
        System.out.println(roleName + " policy is already attached to
this role.");
        return;
    }
}

AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
    .roleName(roleName)
    .policyArn(policyArn)
    .build();

iam.attachRolePolicy(attachRequest);

System.out.println("Successfully attached policy " + policyArn +
    " to role " + roleName);

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

- API の詳細については、「API リファレンス [AttachRolePolicy](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateAccessKey

次の例は、CreateAccessKey を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAccessKey {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <user>\s

                Where:
                user - An AWS IAM user that you can obtain from the AWS
Management Console.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String user = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
```

```
        .region(region)
        .build();

    String keyId = createIAMAccessKey(iam, user);
    System.out.println("The Key Id is " + keyId);
    iam.close();
}

public static String createIAMAccessKey(IamClient iam, String user) {
    try {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(user)
            .build();

        CreateAccessKeyResponse response = iam.createAccessKey(request);
        return response.accessKey().accessKeyId();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- APIの詳細については、「API リファレンス [CreateAccessKey](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateAccountAlias

次の例は、CreateAccountAlias を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.CreateAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAccountAlias {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <alias>\s

            Where:
                alias - The account alias to create (for example, myawsaccount).
\s

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        createIAMAccountAlias(iam, alias);
        iam.close();
        System.out.println("Done");
    }

    public static void createIAMAccountAlias(IamClient iam, String alias) {
        try {
            CreateAccountAliasRequest request = CreateAccountAliasRequest.builder()
```

```
        .accountAlias(alias)
        .build();

        iam.createAccountAlias(request);
        System.out.println("Successfully created account alias: " + alias);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [CreateAccountAlias](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreatePolicy

次の例は、CreatePolicy を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;

/**
 * Before running this Java V2 code example, set up your development
```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreatePolicy {

    public static final String PolicyDocument = "{" +
        "  \"Version\": \"2012-10-17\", " +
        "  \"Statement\": [ " +
        "    { " +
        "      \"Effect\": \"Allow\", " +
        "      \"Action\": [ " +
        "        \"dynamodb:DeleteItem\", " +
        "        \"dynamodb:GetItem\", " +
        "        \"dynamodb:PutItem\", " +
        "        \"dynamodb:Scan\", " +
        "        \"dynamodb:UpdateItem\" " +
        "      ], " +
        "      \"Resource\": \"*\":" +
        "    } " +
        "  ] " +
        "}";

    public static void main(String[] args) {

        final String usage = ""
            Usage:
              CreatePolicy <policyName>\s

        Where:
          policyName - A unique policy name.\s
        """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyName = args[0];
        Region region = Region.AWS_GLOBAL;
        IAMClient iam = IAMClient.builder()
            .region(region)
```



```
        .build();

        String result = createIAMPolicy(iam, policyName);
        System.out.println("Successfully created a policy with this ARN value: " +
result);
        iam.close();
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreatePolicyRequest request = CreatePolicyRequest.builder()
                .policyName(policyName)
                .policyDocument(PolicyDocument)
                .build();

            CreatePolicyResponse response = iam.createPolicy(request);

            // Wait until the policy is created.
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                .policyArn(response.policy().arn())
                .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

            waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- APIの詳細については、「APIリファレンス[CreatePolicy](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateRole

次の例は、CreateRole を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import software.amazon.awssdk.services.iam.model.CreateRoleRequest;
import software.amazon.awssdk.services.iam.model.CreateRoleResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import java.io.FileReader;

/*
 * This example requires a trust policy document. For more information, see:
 * https://aws.amazon.com/blogs/security/how-to-use-trust-policies-with-iam-roles/
 *
 * In addition, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateRole {
    public static void main(String[] args) throws Exception {
        final String usage = ""
            Usage:
                <rolename> <fileLocation>\s

            Where:
                rolename - The name of the role to create.\s
                fileLocation - The location of the JSON document that represents
the trust policy.\s
```

```
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String rolename = args[0];
    String fileLocation = args[1];
    Region region = Region.AWS_GLOBAL;
    IAMClient iam = IAMClient.builder()
        .region(region)
        .build();

    String result = createIAMRole(iam, rolename, fileLocation);
    System.out.println("Successfully created user: " + result);
    iam.close();
}

public static String createIAMRole(IAMClient iam, String rolename, String
fileLocation) throws Exception {
    try {
        JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(jsonObject.toJSONString())
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is " + response.role().arn());

    } catch (IAMException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static Object readJsonSimpleDemo(String filename) throws Exception {
    FileReader reader = new FileReader(filename);
    JSONParser jsonParser = new JSONParser();
    return jsonParser.parse(reader);
}
```

```
}
```

- API の詳細については、「API リファレンス [CreateRole](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateUser

次の例は、CreateUser を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUser {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
```

```
        <username>\s

        Where:
            username - The name of the user to create.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String username = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    String result = createIAMUser(iam, username);
    System.out.println("Successfully created user: " + result);
    iam.close();
}

public static String createIAMUser(IamClient iam, String username) {
    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();

        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        CreateUserResponse response = iam.createUser(request);

        // Wait until the user is created.
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user().userName();
    } catch (IamException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- API の詳細については、「API リファレンス [CreateUser](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteAccessKey

次の例は、DeleteAccessKey を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAccessKey {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
```

```
        <username> <accessKey>\s

        Where:
            username - The name of the user.\s
            accessKey - The access key ID for the secret access key you want
to delete.\s
        """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessKey = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();
        deleteKey(iam, username, accessKey);
        iam.close();
    }

    public static void deleteKey(IamClient iam, String username, String accessKey) {
        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                .accessKeyId(accessKey)
                .userName(username)
                .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                " from user " + username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [DeleteAccessKey](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteAccountAlias

次の例は、DeleteAccountAlias を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.DeleteAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAccountAlias {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <alias>\s

                Where:
                alias - The account alias to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```



```
    }

    String alias = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    deleteIAMAccountAlias(iam, alias);
    iam.close();
}

public static void deleteIAMAccountAlias(IamClient iam, String alias) {
    try {
        DeleteAccountAliasRequest request = DeleteAccountAliasRequest.builder()
            .accountAlias(alias)
            .build();

        iam.deleteAccountAlias(request);
        System.out.println("Successfully deleted account alias " + alias);


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

- APIの詳細については、「APIリファレンス[DeleteAccountAlias](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeletePolicy

次の例は、DeletePolicyを使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.DeletePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeletePolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <policyARN>\s

            Where:
                policyARN - A policy ARN value to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyARN = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();
```

```
        deleteIAMPolicy(iam, policyARN);
        iam.close();
    }

    public static void deleteIAMPolicy(IamClient iam, String policyARN) {
        try {
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                .policyArn(policyARN)
                .build();

            iam.deletePolicy(request);
            System.out.println("Successfully deleted the policy");

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

- API の詳細については、「API リファレンス [DeletePolicy](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteUser

次の例は、DeleteUser を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
```

```
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteUser {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <userName>\s

                Where:
                userName - The name of the user to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        deleteIAMUser(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                .userName(userName)
                .build();

            iam.deleteUser(request);
        }
    }
}
```

```
        System.out.println("Successfully deleted IAM user " + userName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DeleteUser](#)」の「」を参照してください。AWS SDK for Java 2.x

## DetachRolePolicy

次の例は、DetachRolePolicy を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetachRolePolicy {
    public static void main(String[] args) {
        final String usage = ""
```

```
Usage:
    <roleName> <policyArn>\s

Where:
    roleName - A role name that you can obtain from the AWS
Management Console.\s
    policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
    """;

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String roleName = args[0];
String policyArn = args[1];
Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .build();
detachPolicy(iam, roleName, policyArn);
System.out.println("Done");
iam.close();
}

public static void detachPolicy(IamClient iam, String roleName, String
policyArn) {
    try {
        DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.detachRolePolicy(request);
        System.out.println("Successfully detached policy " + policyArn +
            " from role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- API の詳細については、「API リファレンス [DetachRolePolicy](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListAccessKeys

次の例は、ListAccessKeys を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListAccessKeys {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <userName>\s

                Where:
```

```
        userName - The name of the user for which access keys are
retrieved.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String userName = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    listKeys(iam, userName);
    System.out.println("Done");
    iam.close();
}

public static void listKeys(IamClient iam, String userName) {
    try {
        boolean done = false;
        String newMarker = null;

        while (!done) {
            ListAccessKeysResponse response;

            if (newMarker == null) {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .build();

                response = iam.listAccessKeys(request);
            } else {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .marker(newMarker)
                    .build();

                response = iam.listAccessKeys(request);
            }
        }
    }
}
```



```
        for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
            System.out.format("Retrieved access key %s",
                metadata.accessKeyId());
        }

        if (!response.isTruncated()) {
            done = true;
        } else {
            newMarker = response.marker();
        }
    }

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [ListAccessKeys](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListAccountAliases

次の例は、ListAccountAliases を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccountAliasesResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListAccountAliases {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listAliases(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAliases(IamClient iam) {
        try {
            ListAccountAliasesResponse response = iam.listAccountAliases();
            for (String alias : response.accountAliases()) {
                System.out.printf("Retrieved account alias %s", alias);
            }
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- APIの詳細については、「APIリファレンス[ListAccountAliases](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListUsers

次の例は、ListUsers を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.AttachedPermissionsBoundary;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.User;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUsers {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listAllUsers(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAllUsers(IamClient iam) {
        try {
            boolean done = false;
            String newMarker = null;
            while (!done) {
                ListUsersResponse response;
```

```
        if (newMarker == null) {
            ListUsersRequest request = ListUsersRequest.builder().build();
            response = iam.listUsers(request);
        } else {
            ListUsersRequest request = ListUsersRequest.builder()
                .marker(newMarker)
                .build();

            response = iam.listUsers(request);
        }

        for (User user : response.users()) {
            System.out.format("\n Retrieved user %s", user.userName());
            AttachedPermissionsBoundary permissionsBoundary =
user.permissionsBoundary();
            if (permissionsBoundary != null)
                System.out.format("\n Permissions boundary details %s",
permissionsBoundary.permissionsBoundaryTypeAsString());
        }

        if (!response.isTruncated()) {
            done = true;
        } else {
            newMarker = response.marker();
        }
    }

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス [ListUsers](#)」の「」を参照してください。AWS SDK for Java 2.x

## UpdateAccessKey

次の例は、UpdateAccessKey を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateAccessKey {

    private static StatusType statusType;

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <username> <accessId> <status>\s

            Where:
                username - The name of the user whose key you want to update.\s
                accessId - The access key ID of the secret access key you want
to update.\s
                status - The status you want to assign to the secret access key.
\s

            """;

        if (args.length != 3) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String username = args[0];
    String accessId = args[1];
    String status = args[2];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    updateKey(iam, username, accessId, status);
    System.out.println("Done");
    iam.close();
}

public static void updateKey(IamClient iam, String username, String accessId,
String status) {
    try {
        if (status.toLowerCase().equalsIgnoreCase("active")) {
            statusType = StatusType.ACTIVE;
        } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
            statusType = StatusType.INACTIVE;
        } else {
            statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
        }

        UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
            .accessKeyId(accessId)
            .userName(username)
            .status(statusType)
            .build();

        iam.updateAccessKey(request);
        System.out.printf("Successfully updated the status of access key %s to"
+
            "status %s for user %s", accessId, status, username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [UpdateAccessKey](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## UpdateUser

次の例は、UpdateUser を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateUser {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <curName> <newName>\s

                Where:
                curName - The current user name.\s
                newName - An updated user name.\s
        """;
```

```
    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String curName = args[0];
    String newName = args[1];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    updateIAMUser(iam, curName, newName);
    System.out.println("Done");
    iam.close();
}

public static void updateIAMUser(IamClient iam, String curName, String newName)
{
    try {
        UpdateUserRequest request = UpdateUserRequest.builder()
            .userName(curName)
            .newUserName(newName)
            .build();

        iam.updateUser(request);
        System.out.printf("Successfully updated user to username %s", newName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [UpdateUser](#)」の「」を参照してください。AWS SDK for Java 2.x



## シナリオ

### レジリエントなサービスの構築と管理

次のコード例は、本、映画、曲のレコメンデーションを返す負荷分散型ウェブサービスの作成方法を示しています。この例は、障害に対するサービスの対応方法と、障害発生時の耐障害性を高めるためにサービスを再構築する方法を示しています。

- Amazon EC2 Auto Scaling グループを使用して、起動テンプレートに基づいて Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを作成し、インスタンス数を所定の範囲内に維持します。
- Elastic Load Balancing で HTTP リクエストを処理して配信します。
- Auto Scaling グループ内のインスタンスの状態を監視し、正常なインスタンスにのみリクエストを転送します。
- 各 EC2 インスタンスで Python ウェブサーバーを実行して HTTP リクエストを処理します。ウェブサーバーはレコメンデーションとヘルスチェックを返します。
- Amazon DynamoDB テーブルを使用してレコメンデーションサービスをシミュレートできます。
- AWS Systems Manager パラメータを更新して、リクエストとヘルスチェックに対するウェブサーバーの応答を制御します。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
public class Main {  
  
    public static final String fileName = "C:\\AWS\\resworkflow\\  
\\recommendations.json"; // Modify file location.  
    public static final String tableName = "doc-example-recommendation-service";  
    public static final String startScript = "C:\\AWS\\resworkflow\\  
\\server_startup_script.sh"; // Modify file location.  
  
}
```

```
    public static final String policyFile = "C:\\\\AWS\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\\\AWS\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
    {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
    }
}
```

```
System.out.println(DASHES);
System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
System.out.println("Press Enter when you're ready.");
in.nextLine();
demo(loadBalancer);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("C - DELETE THE RESOURCES");
System.out.println("""
    This concludes the demo of how to build and manage a resilient
service.

    To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
    that were created for this demo.
    """);

System.out.println("\n Do you want to delete the resources (y/n)? ");
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

if (userInput.equals("y")) {
    // Delete resources here
    deleteResources(loadBalancer, autoScaler, database);
    System.out.println("Resources deleted.");
} else {
    System.out.println("""
        Okay, we'll leave the resources intact.
        Don't forget to delete them when you're done with them or you
might incur unexpected charges.
        """);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The example has completed. ");
System.out.println("\n Thanks for watching!");
System.out.println(DASHES);
}

// Deletes the AWS resources used in this example.
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
    throws IOException, InterruptedException {
    loadBalancer.deleteLoadBalancer(lbName);
```

```
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScalingGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
    InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
            """
                For this demo, we'll use the AWS SDK for Java (v2) to create
    several AWS resources
                to set up a load-balanced web service endpoint and explore
    some ways to make it resilient
                against various kinds of failures.

                Some of the resources create by this demo are:
                \t* A DynamoDB table that the web service depends on to
    provide book, movie, and song recommendations.
                \t* An EC2 launch template that defines EC2 instances that
    each contain a Python web server.
                \t* An EC2 Auto Scaling group that manages EC2 instances
    across several Availability Zones.
                \t* An Elastic Load Balancing (ELB) load balancer that
    targets the Auto Scaling group to distribute requests.
            """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
    tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
```

Creating an EC2 launch template that runs '{startup\_script}' when an instance starts.

This script starts a Python web server defined in the `server.py` script. The web server

listens to HTTP requests on port 80 and responds to requests to '/' and to '/healthcheck'.

For demo purposes, this server is run as the root user. In production, the best practice is to run a web server, such as Apache, with least-privileged credentials.

The template also defines an IAM policy that each instance uses to assume a role that grants permissions to access the DynamoDB recommendation table and Systems Manager parameters that control the flow of the demo.

```
""");
```

```
LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println(
    "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
System.out.println("*** Wait 30 secs for the VPC to be created");
TimeUnit.SECONDS.sleep(30);
AutoScaler autoScaler = new AutoScaler();
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);
```

```
System.out.println("""
    At this point, you have EC2 instances created. Once each instance
starts, it listens for
    HTTP requests. You can see these instances in the console or
continue with the demo.
    Press Enter when you're ready to continue.
""");
```

```
in.nextLine();
System.out.println(DASHES);
```

```
System.out.println(DASHES);
```

```
System.out.println("Creating variables that control the flow of the demo.");
ParameterHelper paramHelper = new ParameterHelper();
paramHelper.reset();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an Elastic Load Balancing target group and load balancer.
The target group
    defines how the load balancer connects to instances. The load
balancer provides a
    single endpoint where clients connect and dispatches requests to
instances in the group.
    """);

String vpcId = autoScaler.getDefaultVPC();
List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
System.out.println("Verifying access to the load balancer endpoint...");
boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
if (!wasSuccessful) {
    System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to "http://checkip.amazonaws.com"
    HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
    try {
        // Execute the request and get the response
        HttpResponse response = httpClient.execute(httpGet);

        // Read the response content.
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

        // Print the public IP address.
        System.out.println("Public IP Address: " + ipAddress);
```

```
        GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
        if (!groupInfo.isPortOpen()) {
            System.out.println("""
                For this example to work, the default security group for
your default VPC must
                allow access from this computer. You can either add it
automatically from this
                example or add it yourself using the AWS Management
Console.
                """);

            System.out.println(
                "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
            System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
            String ans = in.nextLine();
            if ("y".equalsIgnoreCase(ans)) {
                autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                System.out.println("Security group rule added.");
            } else {
                System.out.println("No security group rule added.");
            }
        }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
} else if (wasSuccessful) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" + elbDnsName);
} else {
    System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
    System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
    System.out.println("you can successfully make a GET request to the load
balancer.");
}
```

```
        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
            """
                This part of the demonstration shows how to toggle
different parts of the system
                to create situations where the web service fails, and shows
how using a resilient
                architecture can keep the web service running in spite of
these failures.

                At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
            """);
        demoChoices(loadBalancer);

        System.out.println(
            """
                The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
            """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
            """
                \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
```



```
        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
        """);
demoChoices(loadBalancer);

System.out.println(
    ""
        Instead of failing when the recommendation service fails,
the web service can return a static response.
        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
        """);
paramHelper.put(paramHelper.failureResponse, "static");

System.out.println("""
    Now, sending a GET request to the load balancer endpoint returns a
static response.
    The service still reports as healthy because health checks are still
shallow.
        """);
demoChoices(loadBalancer);

System.out.println("Let's reinstate the recommendation service.");
paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

System.out.println("""
    Let's also substitute bad credentials for one of the instances in
the target group so that it can't
    access the DynamoDB recommendation table. We will get an instance id
value.
        """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
AutoScaler autoScaler = new AutoScaler();

// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " + profileAssociationId);
```

```
System.out.println("Replacing the profile for instance " + badInstanceId
    + " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
    ""
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
    "");

demoChoices(loadBalancer);

System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
        the web service can access the DynamoDB table that it depends on for
recommendations. Note that
        the deep health check is only for ELB routing and not for Auto
Scaling instance health.
        This kind of deep health check is not recommended for Auto Scaling
instance health, because it
        risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
    """);

System.out.println("""
    By implementing deep health checks, the load balancer can detect
when one of the instances is failing
        and take that instance out of rotation.
    """);

paramHelper.put(paramHelper.healthCheck, "deep");

System.out.println("""
    Now, checking target health indicates that the instance with bad
credentials
        is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
        instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
        the load balancer takes unhealthy instances out of its rotation.
    """);
```

```
        """);

        demoChoices(loadBalancer);

        System.out.println(
            ""
                "Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                instance is to terminate it and let the auto scaler start a
new instance to replace it.
                """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
            Even while the instance is terminating and the new instance is
starting, sending a GET
            request to the web service continues to get a successful
recommendation response because
            the load balancer routes requests to the healthy instances. After
the replacement instance
            starts and reports as healthy, it is included in the load balancing
rotation.

            Note that terminating and replacing an instance typically takes
several minutes, during which time you
            can see the changing health check status until the new instance is
running and healthy.
            """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        }
    }
}
```

```
};
Scanner scanner = new Scanner(System.in);

while (true) {
    System.out.println("-".repeat(88));
    System.out.println("See the current state of the service by selecting
one of the following choices:");
    for (int i = 0; i < actions.length; i++) {
        System.out.println(i + ": " + actions[i]);
    }

    try {
        System.out.print("\nWhich action would you like to take? ");
        int choice = scanner.nextInt();
        System.out.println("-".repeat(88));

        switch (choice) {
            case 0 -> {
                System.out.println("Request:\n");
                System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                CloseableHttpClient httpClient =
HttpClientClients.createDefault();

                // Create an HTTP GET request to the ELB.
                HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);

                // Display the JSON response
                BufferedReader reader = new BufferedReader(
                    new
InputStreamReader(response.getEntity().getContent()));
                StringBuilder jsonResponse = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null) {
                    jsonResponse.append(line);
                }
                reader.close();
            }
        }
    }
}
```

```
        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();

    }
    case 1 -> {
        System.out.println("\nChecking the health of load balancer
targets:\n");

        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value between
0-2. Please select again.");
}

} catch (java.util.InputMismatchException e) {
    System.out.println("Invalid input. Please select again.");
    scanner.nextLine(); // Clear the input buffer.
}
}
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
```

```
}  
}
```

Auto Scaling と Amazon EC2 のアクションをラップするクラスを作成します。

```
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.getInstanceProfile().getInstanceProfileName();
        System.out.println(name);

        RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .roleName(roleName)
            .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
            .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(attachedPolicy.policyArn())
                .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");
```

```
        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                .name("group-name")
                .values("default")

```

```
        .build();

    Filter filter1 = Filter.builder()
        .name("vpc-id")
        .values(VPC)
        .build();

    DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
        .filters(filter, filter1)
        .build();

    DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
        .describeSecurityGroups(securityGroupsRequest);
    String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
    groupInfo.setGroupName(securityGroup);

    for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
        System.out.println("Found security group: " + secGroup.groupId());

        for (IpPermission ipPermission : secGroup.ipPermissions()) {
            if (ipPermission.fromPort() == port) {
                System.out.println("Found inbound rule: " + ipPermission);
                for (IpRange ipRange : ipPermission.ipRanges()) {
                    String cidrIp = ipRange.cidrIp();
                    if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                        System.out.println(cidrIp + " is applicable");
                        portIsOpen = true;
                    }
                }

                if (!ipPermission.prefixListIds().isEmpty()) {
                    System.out.println("Prefix lList is applicable");
                    portIsOpen = true;
                }

                if (!portIsOpen) {
                    System.out
                        .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                }
            }
        }
    }
}
```

```
        } else {
            break;
        }
    }
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/*
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {
```

```
// Get availability zones.
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
    .builder()
    .build();

DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

String availabilityZones = String.join(",", availabilityZoneNames);
LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

String[] zones = availabilityZones.split(",");
CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

try {
    getAutoScalingClient().createAutoScalingGroup(groupRequest);

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
return zones;
}
```

```
public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();

    DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
        .filters(vpcFilter, azFilter, defaultForAZ)
        .build();

    DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
    subnets = response.subnets();
    return subnets;
}
```

```
// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
```



```

        ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                .builder()
                .policyArn(policy.arn())
                .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .roleName(roleName) // Specify the name of the IAM role
                        .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
                    .policyArn(policy.arn())
                    .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

// List the roles associated with the instance profile

```

```
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .roleName(roleName) // Remove the extra dot here
        .build();

    getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
    System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
}

// Delete the instance profile after removing all roles
DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
    .build();

getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
System.out.println(InstanceProfile + " Deleted");
System.out.println("All roles and policies are deleted.");
}
}
```

Elastic Load Balancing のアクションをラップするクラスを作成します。

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
```

```
        .region(Region.US_EAST_1)
        .build();
    }

    return elasticLoadBalancingV2Client;
}

// Checks the health of the instances in the target group.
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
        .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
```

```

        .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
        .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {

```

```
        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);
        if (statusCode == 200) {
            success = true;
        } else {
            retries--;
            System.out.println("Got connection error from load balancer
endpoint, retrying...");
            TimeUnit.SECONDS.sleep(15);
        }
    }

} catch (org.apache.http.conn.HttpHostConnectException e) {
    System.out.println(e.getMessage());
}

System.out.println("Status.." + success);
return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
}
```

```
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
                .scheme("internet-facing")
                .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                .loadBalancerArns(lbARN)
                .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                .waitUntilLoadBalancerAvailable(request);
```

```
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Load Balancer " + lbName + " is available.");

// Get the DNS name (endpoint) of the load balancer.
String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

// Create a listener for the load balance.
Action action = Action.builder()
    .targetGroupArn(targetGroupARN)
    .type("forward")
    .build();

CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
    .defaultActions(action)
    .port(port)
    .protocol(protocol)
    .defaultActions(action)
    .build();

getLoadBalancerClient().createListener(listenerRequest);
System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
    + targetGroupARN);

// Return the load balancer DNS name.
return lbDNSName;

} catch (ElasticLoadBalancingV2Exception e) {
    e.printStackTrace();
}
return "";
}
}
```

DynamoDB を使用してレコメンデーションサービスをシミュレートするクラスを作成します。

```
public class Database {

    private static DynamoDbClient dynamoDbClient;
```

```
public static DynamoDbClient getDynamoDbClient() {
    if (dynamoDbClient == null) {
        dynamoDbClient = DynamoDbClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return dynamoDbClient;
}

// Checks to see if the Amazon DynamoDB table exists.
private boolean doesTableExist(String tableName) {
    try {
        // Describe the table and catch any exceptions.
        DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        getDynamoDbClient().describeTable(describeTableRequest);
        System.out.println("Table '" + tableName + "' exists.");
        return true;
    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " + e.getMessage());
    }
    return false;
}

/**
 * Creates a DynamoDB table to use a recommendation service. The table has a
 * hash key named 'MediaType' that defines the type of media recommended, such
 * as
 * Book or Movie, and a range key named 'ItemId' that, combined with the
 * MediaType,
 * forms a unique identifier for the recommended item.
 */
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
    }
}
```



```
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
            .provisionedThroughput(
                ProvisionedThroughput.builder()
                    .readCapacityUnits(5L)
                    .writeCapacityUnits(5L)
                    .build())
            .build();

        getDynamoDbClient().createTable(createTableRequest);
        System.out.println("Creating table " + tableName + "...");

        // Wait until the Amazon DynamoDB table is created.
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitForTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Table " + tableName + " created.");

        // Add records to the table.
        populateTable(fileName, tableName);
    }
}
```

```
public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
    System.out.println("Added all records to the " + tableName);
}
}
```

Systems Manager のアクションをラップするクラスを作成します。

```
public class ParameterHelper {
```

```
String tableName = "doc-example-resilient-architecture-table";
String dyntable = "doc-example-recommendation-service";
String failureResponse = "doc-example-resilient-architecture-failure-response";
String healthCheck = "doc-example-resilient-architecture-health-check";

public void reset() {
    put(dyntable, tableName);
    put(failureResponse, "none");
    put(healthCheck, "shallow");
}

public void put(String name, String value) {
    SsmClient ssmClient = SsmClient.builder()
        .region(Region.US_EAST_1)
        .build();

    PutParameterRequest parameterRequest = PutParameterRequest.builder()
        .name(name)
        .value(value)
        .overwrite(true)
        .type("String")
        .build();


    ssmClient.putParameter(parameterRequest);
    System.out.printf("Setting demo parameter %s to '%s'.", name, value);
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [AttachLoadBalancerTargetGroups](#)
  - [CreateAutoScalingGroup](#)
  - [CreateInstanceProfile](#)
  - [CreateLaunchTemplate](#)
  - [CreateListener](#)
  - [CreateLoadBalancer](#)
  - [CreateTargetGroup](#)
  - [DeleteAutoScalingGroup](#)

- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacesIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

ユーザーを作成してロールを引き受ける

次のコードサンプルは、ユーザーを作成してロールを割り当てる方法を示しています。

 Warning

セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、[AWS IAM Identity Center](#)などの ID プロバイダーとのフェデレーションを使用してください。

- 権限のないユーザーを作成します。
- 指定したアカウントに Amazon S3 バケットへのアクセス権限を付与するロールを作成します。
- ユーザーにロールを引き受けさせるポリシーを追加します。

- ロールを引き受け、一時的な認証情報を使用して S3 バケットを一覧表示しリソースをクリーンアップします。

## SDK for Java 2.x

### Note

については、「」を参照してください GitHub。用例一覧を検索し、[AWS コード例リポジトリ](#)での設定と実行の方法を確認してください。

IAM ユーザーアクションをラップする関数を作成します。

```
/*
  To run this Java V2 code example, set up your development environment, including
  your credentials.

  For information, see this documentation topic:

  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

  This example performs these operations:

  1. Creates a user that has no permissions.
  2. Creates a role and policy that grants Amazon S3 permissions.
  3. Creates a role.
  4. Grants the user permissions.
  5. Gets temporary credentials by assuming the role.  Creates an Amazon S3 Service
  client object with the temporary credentials.
  6. Deletes the resources.
*/

public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static final String PolicyDocument = "{" +
        "  \"Version\": \"2012-10-17\",\" +
        "  \"Statement\": [\" +
        "    {\" +
        "      \"Effect\": \"Allow\",\" +
        "      \"Action\": [\" +
        "        \"s3:*\"\" +
        "      ],\" +
```

```
        "        \"Resource\": \"*\")\" +
        \"    }\" +
        \"    ]\" +
        \"}\";

public static String userArn;

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

    final String usage = \"\"\"

        Usage:
            <username> <policyName> <roleName> <roleSessionName>
<bucketName>\\s

        Where:
            username - The name of the IAM user to create.\\s
            policyName - The name of the policy to create.\\s
            roleName - The name of the role to create.\\s
            roleSessionName - The name of the session required for the
assumeRole operation.\\s
            bucketName - The name of the Amazon S3 bucket from which objects
are read.\\s

        \"\"\";

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String userName = args[0];
    String policyName = args[1];
    String roleName = args[2];
    String roleSessionName = args[3];
    String bucketName = args[4];

    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println(\"Welcome to the AWS IAM example scenario.\");
    System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println(" 1. Create the IAM user.");
User createUser = createIAMUser(iam, userName);

System.out.println(DASHES);
userArn = createUser.arn();

AccessKey myKey = createIAMAccessKey(iam, userName);
String accessKey = myKey.accessKeyId();
String secretKey = myKey.secretAccessKey();
String assumeRolePolicyDocument = "{" +
    "\"Version\": \"2012-10-17\"," +
    "\"Statement\": [{" +
    "\"Effect\": \"Allow\"," +
    "\"Principal\": {" +
    "  \"AWS\": \"" + userArn + "\"" +
    "}," +
    "\"Action\": \"sts:AssumeRole\"" +
    "}]}" +
    "}";

System.out.println(assumeRolePolicyDocument);
System.out.println(userName + " was successfully created.");
System.out.println(DASHES);
System.out.println("2. Creates a policy.");
String polArn = createIAMPolicy(iam, policyName);
System.out.println("The policy " + polArn + " was successfully created.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Creates a role.");
TimeUnit.SECONDS.sleep(30);
String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
System.out.println(roleArn + " was successfully created.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Grants the user permissions.");
attachIAMRolePolicy(iam, roleName, polArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("*** Wait for 30 secs so the resource is available");
```

```
        TimeUnit.SECONDS.sleep(30);
        System.out.println("5. Gets temporary credentials by assuming the role.");
        System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
        assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6 Getting ready to delete the AWS resources");
        deleteKey(iam, userName, accessKey);
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                .userName(user)
                .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static User createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();
            CreateUserRequest request = CreateUserRequest.builder()
                .userName(username)
                .build();

            // Wait until the user is created.
```



```
        CreateUserResponse response = iam.createUser(request);
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static String createIAMRole(IamClient iam, String rolename, String json)
{

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(json)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is " + response.role().arn());
        return response.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createIAMPolicy(IamClient iam, String policyName) {
    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();
        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
```

```
        .policyDocument(PolicyDocument).build());

    CreatePolicyResponse response = iam.createPolicy(request);
    GetPolicyRequest polRequest = GetPolicyRequest.builder()
        .policyArn(response.policy().arn())
        .build();

    WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
    return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
    try {
        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
            .roleName(roleName)
            .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
        String polArn;
        for (AttachedPolicy policy : attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn) == 0) {
                System.out.println(roleName + " policy is already attached to
this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
            .roleName(roleName)
```

```
        .policyArn(policyArn)
        .build();

    iam.attachRolePolicy(attachRequest);
    System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Invoke an Amazon S3 operation using the Assumed Role.
public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
    String keySecret) {

    // Use the creds of the new IAM user that was created in this code example.
    AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
    StsClient stsClient = StsClient.builder()
        .region(Region.US_EAST_1)
        .credentialsProvider(StaticCredentialsProvider.create(credentials))
        .build();

    try {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
            .roleArn(roleArn)
            .roleSessionName(roleSessionName)
            .build();

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        Credentials myCreds = roleResponse.credentials();
        String key = myCreds.accessKeyId();
        String secKey = myCreds.secretAccessKey();
        String secToken = myCreds.sessionToken();

        // List all objects in an Amazon S3 bucket using the temp creds
retrieved by
        // invoking assumeRole.
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .credentialsProvider(
```

```
StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
secToken)))
        .region(region)
        .build();

System.out.println("Created a S3Client using temp credentials.");
System.out.println("Listing objects in " + bucketName);
ListObjectsRequest listObjects = ListObjectsRequest.builder()
        .bucket(bucketName)
        .build();

ListObjectsResponse res = s3.listObjects(listObjects);
List<S3Object> objects = res.contents();
for (S3Object myValue : objects) {
    System.out.println("The name of the key is " + myValue.key());
    System.out.println("The owner is " + myValue.owner());
}

} catch (StsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void deleteRole(IamClient iam, String roleName, String polArn) {

    try {
        // First the policy needs to be detached.
        DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
            .policyArn(polArn)
            .roleName(roleName)
            .build();

        iam.detachRolePolicy(rolePolicyRequest);

        // Delete the policy.
        DeletePolicyRequest request = DeletePolicyRequest.builder()
            .policyArn(polArn)
            .build();

        iam.deletePolicy(request);
        System.out.println("*** Successfully deleted " + polArn);
    }
}
```

```
// Delete the role.
DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
    .roleName(roleName)
    .build();

iam.deleteRole(roleRequest);
System.out.println("*** Successfully deleted " + roleName);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void deleteKey(IamClient iam, String username, String accessKey) {
    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteIAMUser(IamClient iam, String userName) {
    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("*** Successfully deleted " + userName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## IAM Policy Builder API を使用する

次のコードサンプルは、以下の操作方法を示しています。

- オブジェクト指向の API を使用して IAM ポリシーを作成します。
- IAM サービスで IAM Policy Builder API を使用します。

## SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

例では、次の入力を使用します。

```
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;
```

時間ベースのポリシーを作成します。

```
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());
}

```

複数の条件を含むポリシーを作成します。

```

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());
}
```

ポリシーにプリンシパルを使用します。

```
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::4444455556666:user/user-name"))
        .build();
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}
```

クロスアカウントのアクセスを許可します。

```
public String allowCrossAccountAccessExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
```

```

        "111122223333")
        .effect(IamEffect.ALLOW)
        .addPrincipal(IamPrincipalType.AWS,
        EXAMPLE-BUCKET/*")
        .addAction("s3:PutObject")
        .addResource("arn:aws:s3::DOC-
        owner-full-control"))
        .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());
}

```

IamPolicy を作成してアップロードします。

```

    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());
    }
}

```

IamPolicy をダウンロードして使用します。

```
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());
    }
```

- 詳細については、「[AWS SDK for Java 2.x デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [CreatePolicy](#)
  - [GetPolicy](#)
  - [GetPolicyVersion](#)

## AWS IoT SDK for Java 2.x を使用した の例

次のコード例は、AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS IoT。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### 開始方法

こんにちは AWS IoTは

次のコード例は、AWS IoTの使用を開始する方法を示しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
import java.util.List;

public class HelloIoT {
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllThings(iotClient);
    }

    public static void listAllThings( IotClient iotClient) {
        ListThingsRequest thingsRequest = ListThingsRequest.builder()
            .maxResults(10)
            .build();

        ListThingsResponse response = iotClient.listThings(thingsRequest) ;
        List<ThingAttribute> thingList = response.things();
        for (ThingAttribute attribute : thingList) {
            System.out.println("Thing name: "+attribute.thingName());
            System.out.println("Thing ARN: "+attribute.thingArn());
        }
    }
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[listThings](#)」を参照してください。

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### AttachThingPrincipal

次の例は、AttachThingPrincipal を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void attachCertificateToThing(IotClient iotClient, String
thingName, String certificateArn) {
    // Attach the certificate to the thing.
    AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
    .thingName(thingName)
    .principal(certificateArn)
    .build();

    AttachThingPrincipalResponse attachResponse =
iotClient.attachThingPrincipal(principalRequest);

    // Verify the attachment was successful.
    if (attachResponse.sdkHttpResponse().isSuccessful()) {
        System.out.println("Certificate attached to Thing successfully.");

        // Print additional information about the Thing.
        describeThing(iotClient, thingName);
    } else {
        System.err.println("Failed to attach certificate to Thing. HTTP Status
Code: " +
            attachResponse.sdkHttpResponse().statusCode());
    }
}
```

- API の詳細については、「API リファレンス [AttachThingPrincipal](#)」の「」を参照してください。 AWS SDK for Java 2.x

## CreateKeysAndCertificate

次の例は、CreateKeysAndCertificate を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String createCertificate(IotClient iotClient) {
    try {
        CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
        String certificatePem = response.certificatePem();
        String certificateArn = response.certificateArn();

        // Print the details.
        System.out.println("\nCertificate:");
        System.out.println(certificatePem);
        System.out.println("\nCertificate ARN:");
        System.out.println(certificateArn);
        return certificateArn;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

- API の詳細については、「API リファレンス [CreateKeysAndCertificate](#)」の「」を参照してください。 AWS SDK for Java 2.x

## CreateThing

次の例は、CreateThing を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void createIoTThing(IotClient iotClient, String thingName) {
    try {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CreateThingResponse createThingResponse =
            iotClient.createThing(createThingRequest);
        System.out.println(thingName + " was successfully created. The ARN value
            is " + createThingResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [CreateThing](#)」の「」を参照してください。 AWS SDK for Java 2.x

**CreateTopicRule**

次の例は、CreateTopicRule を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。



```
public static void createIoTRule(IotClient iotClient, String roleARN, String
ruleName, String action) {
    try {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        // Create the rule.
        iotClient.createTopicRule(topicRuleRequest);
        System.out.println("IoT Rule created successfully.");

    } catch (IotException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [CreateTopicRule](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteCertificate

次の例は、DeleteCertificate を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void deleteCertificate(IotClient iotClient, String
certificateArn ) {
    DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
        .certificateId(extractCertificateId(certificateArn))
        .build();

    iotClient.deleteCertificate(certificateProviderRequest);
    System.out.println(certificateArn + " was successfully deleted.");
}
```

- API の詳細については、「API リファレンス [DeleteCertificate](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteThing

次の例は、DeleteThing を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void deleteIoTThing(IotClient iotClient, String thingName) {
```

```
try {
    DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
        .thingName(thingName)
        .build();

    iotClient.deleteThing(deleteThingRequest);
    System.out.println("Deleted Thing " + thingName);

} catch (IotException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス [DeleteThing](#)」の「」を参照してください。AWS SDK for Java 2.x

## DescribeEndpoint

次の例は、DescribeEndpoint を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String describeEndpoint(IotClient iotClient) {
    try {
        DescribeEndpointResponse endpointResponse =
            iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

        // Get the endpoint URL.
        String endpointUrl = endpointResponse.endpointAddress();
        String exString = getValue(endpointUrl);
        String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";
    }
}
```

```
        System.out.println("Full Endpoint URL: "+fullEndpoint);
        return fullEndpoint;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}
```

- API の詳細については、「API リファレンス [DescribeEndpoint](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DescribeThing

次の例は、DescribeThing を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
private static void describeThing(IotClient iotClient, String thingName) {
    try {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build() ;

        // Print Thing details.
        DescribeThingResponse describeResponse =
iotClient.describeThing(thingRequest);
        System.out.println("Thing Details:");
        System.out.println("Thing Name: " + describeResponse.thingName());
        System.out.println("Thing ARN: " + describeResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DescribeThing](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DetachThingPrincipal

次の例は、DetachThingPrincipal を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void detachThingPrincipal(IotClient iotClient, String thingName,
String certificateArn){
    try {
        DetachThingPrincipalRequest thingPrincipalRequest =
        DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        iotClient.detachThingPrincipal(thingPrincipalRequest);
        System.out.println(certificateArn + " was successfully removed from "
+thingName);

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DetachThingPrincipal](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListCertificates

次の例は、ListCertificates を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void listCertificates(IotClient iotClient) {
    ListCertificatesResponse response = iotClient.listCertificates();
    List<Certificate> certList = response.certificates();
    for (Certificate cert : certList) {
        System.out.println("Cert id: " + cert.certificateId());
        System.out.println("Cert Arn: " + cert.certificateArn());
    }
}
```

- API の詳細については、「API リファレンス [ListCertificates](#)」の「」を参照してください。AWS SDK for Java 2.x

## SearchIndex

次の例は、SearchIndex を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void searchThings(IotClient iotClient, String queryString){
    SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
        .queryString(queryString)
        .build();

    try {
        // Perform the search and get the result.
        SearchIndexResponse searchIndexResponse =
iotClient.searchIndex(searchIndexRequest);

        // Process the result.
        if (searchIndexResponse.things().isEmpty()) {
            System.out.println("No things found.");
        } else {
            searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
        }
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- APIの詳細については、「APIリファレンス[SearchIndex](#)」の「」を参照してください。AWS SDK for Java 2.x

## UpdateThing

次の例は、UpdateThing を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void updateThing(IotClient iotClient, String thingName) {
    // Specify the new attribute values.
```

```
String newLocation = "Office";
String newFirmwareVersion = "v2.0";

Map<String, String> attMap = new HashMap<>();
attMap.put("location", newLocation);
attMap.put("firmwareVersion", newFirmwareVersion);

AttributePayload attributePayload = AttributePayload.builder()
    .attributes(attMap)
    .build();

UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
    .thingName(thingName)
    .attributePayload(attributePayload)
    .build();

try {
    // Update the IoT Thing attributes.
    iotClient.updateThing(updateThingRequest);
    System.out.println("Thing attributes updated successfully.");
} catch (IotException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- APIの詳細については、「APIリファレンス[UpdateThing](#)」の「」を参照してください。AWS SDK for Java 2.x

## シナリオ

### デバイス管理のユースケースの使用

次のコード例は、AWS IoT SDK を使用して AWS IoT デバイス管理のユースケースを操作する方法を示しています。



## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.Action;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.AttributePayload;
import software.amazon.awssdk.services.iot.model.Certificate;
import software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointResponse;
import software.amazon.awssdk.services.iot.model.DescribeThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeThingResponse;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.IotException;
import software.amazon.awssdk.services.iot.model.ListCertificatesResponse;
import software.amazon.awssdk.services.iot.model.ListTopicRulesRequest;
import software.amazon.awssdk.services.iot.model.ListTopicRulesResponse;
import software.amazon.awssdk.services.iot.model.SearchIndexRequest;
import software.amazon.awssdk.services.iot.model.SearchIndexResponse;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iot.model.UpdateThingRequest;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowResponse;
import software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowRequest;
import java.net.URI;
```

```
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Creates an AWS IoT Thing.
 * 2. Generate and attach a device certificate.
 * 3. Update an AWS IoT Thing with Attributes.
 * 4. Get an AWS IoT Endpoint.
 * 5. List your certificates.
 * 6. Updates the shadow for the specified thing..
 * 7. Write out the state information, in JSON format
 * 8. Creates a rule
 * 9. List rules
 * 10. Search things
 * 11. Detach and delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String TOPIC = "your-iot-topic";
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <roleARN> <snsAction>

                Where:
                    roleARN - The ARN of an IAM role that has permission to work
with AWS IOT.
                    snsAction - An ARN of an SNS topic.
            """;
    }
}
```

```
        """);

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String thingName;
    String ruleName;
    String roleARN = args[0];
    String snsAction = args[1];
    Scanner scanner = new Scanner(System.in);
    IotClient iotClient = IotClient.builder()
        .region(Region.US_EAST_1)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the AWS IoT example workflow.");
    System.out.println("""
        This example program demonstrates various interactions with the AWS
        Internet of Things (IoT) Core service. The program guides you through a series of
        steps,
            including creating an IoT Thing, generating a device certificate,
            updating the Thing with attributes, and so on.
        It utilizes the AWS SDK for Java V2 and incorporates functionality for
        creating and managing IoT Things, certificates, rules,
            shadows, and performing searches. The program aims to showcase AWS IoT
        capabilities and provides a comprehensive example for
            developers working with AWS IoT in a Java environment.

        """);
    System.out.print("Press Enter to continue...");
    scanner.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Create an AWS IoT Thing.");
    System.out.println("""
        An AWS IoT Thing represents a virtual entity in the AWS IoT service that
        can be associated with a physical device.
        """);
    // Prompt the user for input.
    System.out.print("Enter Thing name: ");
    thingName = scanner.nextLine();
```

```
createIoTThing(iotClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Generate a device certificate.");
System.out.println("""
    A device certificate performs a role in securing the communication
between devices (Things) and the AWS IoT platform.
    """);

System.out.print("Do you want to create a certificate for " +thingName +"?
(y/n)");
String certAns = scanner.nextLine();
String certificateArn="" ;
if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
    certificateArn = createCertificate(iotClient);
    System.out.println("Attach the certificate to the AWS IoT Thing.");
    attachCertificateToThing(iotClient, thingName, certificateArn);
} else {
    System.out.println("A device certificate was not created.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Update an AWS IoT Thing with Attributes.");
System.out.println("""
    IoT Thing attributes, represented as key-value pairs, offer a pivotal
advantage in facilitating efficient data
    management and retrieval within the AWS IoT ecosystem.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
updateThing(iotClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Return a unique endpoint specific to the Amazon Web
Services account.");
System.out.println("""
    An IoT Endpoint refers to a specific URL or Uniform Resource Locator
that serves as the entry point for communication between IoT devices and the AWS
IoT service.
    """);
System.out.print("Press Enter to continue...");
```

```
scanner.nextLine();
String endpointUrl = describeEndpoint(iotClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. List your AWS IoT certificates");
System.out.print("Press Enter to continue...");
scanner.nextLine();
if (certificateArn.length() > 0) {
    listCertificates(iotClient);
} else {
    System.out.println("You did not create a certificates. Skipping this
step.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
System.out.println("""
    A Thing Shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
    of a physical device or thing. The Thing Shadow allows you to
synchronize and control the state of a device between
    the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a Thing Shadow.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
IotDataPlaneClient iotPlaneClient = IotDataPlaneClient.builder()
    .region(Region.US_EAST_1)
    .endpointOverride(URI.create(endpointUrl))
    .build();

updateShadowThing(iotPlaneClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Write out the state information, in JSON format.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
getPayload(iotPlaneClient, thingName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("8. Creates a rule");
System.out.println("""
Creates a rule that is an administrator-level action.
Any user who has permission to create rules will be able to access data
processed by the rule.
""");
System.out.print("Enter Rule name: ");
ruleName = scanner.nextLine();
createIoTRule(iotClient, roleARN, ruleName, snsAction);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. List your rules.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
listIoTRules(iotClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Search things using the Thing name.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
String queryString = "thingName:"+thingName ;
searchThings(iotClient, queryString);
System.out.println(DASHES);

System.out.println(DASHES);
if (certificateArn.length() > 0) {
    System.out.print("Do you want to detach and delete the certificate for "
+thingName +"? (y/n)");
    String delAns = scanner.nextLine();
    if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
        System.out.println("11. You selected to detach amd delete the
certificate.");
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        detachThingPrincipal(iotClient, thingName, certificateArn);
        deleteCertificate(iotClient, certificateArn);
    } else {
        System.out.println("11. You selected not to delete the
certificate.");
    }
} else {
```

```
        System.out.println("11. You did not create a certificate so there is
nothing to delete.");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("12. Delete the AWS IoT Thing.");
    System.out.print("Do you want to delete the IoT Thing? (y/n)");
    String delAns = scanner.nextLine();
    if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
        deleteIoTThing(iotClient, thingName);
    } else {
        System.out.println("The IoT Thing was not deleted.");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The AWS IoT workflow has successfully completed.");
    System.out.println(DASHES);
}

public static void listCertificates(IotClient iotClient) {
    ListCertificatesResponse response = iotClient.listCertificates();
    List<Certificate> certList = response.certificates();
    for (Certificate cert : certList) {
        System.out.println("Cert id: " + cert.certificateId());
        System.out.println("Cert Arn: " + cert.certificateArn());
    }
}

public static void listIoTRules(IotClient iotClient) {
    try {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        ListTopicRulesResponse listTopicRulesResponse =
iotClient.listTopicRules(listTopicRulesRequest);
        System.out.println("List of IoT Rules:");
        List<TopicRuleListItem> ruleList = listTopicRulesResponse.rules();
        for (TopicRuleListItem rule : ruleList) {
            System.out.println("Rule Name: " + rule.ruleName());
            System.out.println("Rule ARN: " + rule.ruleArn());
            System.out.println("-----");
        }
    }
}
```

```
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createIoTRule(IotClient iotClient, String roleARN, String
ruleName, String action) {
    try {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        // Create the rule.
        iotClient.createTopicRule(topicRuleRequest);
        System.out.println("IoT Rule created successfully.");

    } catch (IotException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```



```
public static void getPayload(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        GetThingShadowResponse getThingShadowResponse =
iotPlaneClient.getThingShadow(getThingShadowRequest);

        // Extracting payload from response.
        SdkBytes payload = getThingShadowResponse.payload();
        String payloadString = payload.asUtf8String();
        System.out.println("Received Shadow Data: " + payloadString);

    } catch (IotException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateShadowThing(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}\"";
        SdkBytes data= SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8 );
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        // Update Thing Shadow.
        iotPlaneClient.updateThingShadow(updateThingShadowRequest);
        System.out.println("Thing Shadow updated successfully.");

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}

public static void updateThing(IotClient iotClient, String thingName) {
    // Specify the new attribute values.
    String newLocation = "Office";
    String newFirmwareVersion = "v2.0";

    Map<String, String> attMap = new HashMap<>();
    attMap.put("location", newLocation);
    attMap.put("firmwareVersion", newFirmwareVersion);

    AttributePayload attributePayload = AttributePayload.builder()
        .attributes(attMap)
        .build();

    UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
        .thingName(thingName)
        .attributePayload(attributePayload)
        .build();

    try {
        // Update the IoT Thing attributes.
        iotClient.updateThing(updateThingRequest);
        System.out.println("Thing attributes updated successfully.");
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String describeEndpoint(IotClient iotClient) {
    try {
        DescribeEndpointResponse endpointResponse =
        iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

        // Get the endpoint URL.
        String endpointUrl = endpointResponse.endpointAddress();
        String exString = getValue(endpointUrl);
        String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

        System.out.println("Full Endpoint URL: " + fullEndpoint);
        return fullEndpoint;
    }
}
```

```
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}

public static void detachThingPrincipal(IotClient iotClient, String thingName,
String certificateArn){
    try {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        iotClient.detachThingPrincipal(thingPrincipalRequest);
        System.out.println(certificateArn + " was successfully removed from "
+thingName);

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteCertificate(IotClient iotClient, String
certificateArn ) {
    DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
        .certificateId(extractCertificateId(certificateArn))
        .build();

    iotClient.deleteCertificate(certificateProviderRequest);
    System.out.println(certificateArn + " was successfully deleted.");
}

// Get the cert Id from the Cert ARN value.
private static String extractCertificateId(String certificateArn) {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
    String[] arnParts = certificateArn.split(":");
    String certificateIdPart = arnParts[arnParts.length - 1];
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1);
}
```

```
}

public static String createCertificate(IotClient iotClient) {
    try {
        CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
        String certificatePem = response.certificatePem();
        String certificateArn = response.certificateArn();

        // Print the details.
        System.out.println("\nCertificate:");
        System.out.println(certificatePem);
        System.out.println("\nCertificate ARN:");
        System.out.println(certificateArn);
        return certificateArn;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

public static void attachCertificateToThing(IotClient iotClient, String
thingName, String certificateArn) {
    // Attach the certificate to the thing.
    AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
        .thingName(thingName)
        .principal(certificateArn)
        .build();

    AttachThingPrincipalResponse attachResponse =
iotClient.attachThingPrincipal(principalRequest);

    // Verify the attachment was successful.
    if (attachResponse.sdkHttpResponse().isSuccessful()) {
        System.out.println("Certificate attached to Thing successfully.");

        // Print additional information about the Thing.
        describeThing(iotClient, thingName);
    } else {
```

```
        System.err.println("Failed to attach certificate to Thing. HTTP Status
Code: " +
        attachResponse.sdkHttpResponse().statusCode());
    }
}

private static void describeThing(IotClient iotClient, String thingName) {
    try {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build() ;

        // Print Thing details.
        DescribeThingResponse describeResponse =
iotClient.describeThing(thingRequest);
        System.out.println("Thing Details:");
        System.out.println("Thing Name: " + describeResponse.thingName());
        System.out.println("Thing ARN: " + describeResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteIoTThing(IotClient iotClient, String thingName) {
    try {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        iotClient.deleteThing(deleteThingRequest);
        System.out.println("Deleted Thing " + thingName);

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createIoTThing(IotClient iotClient, String thingName) {
    try {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
```

```
        .build();

        CreateThingResponse createThingResponse =
iotClient.createThing(createThingRequest);
        System.out.println(thingName + " was successfully created. The ARN value
is " + createThingResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

private static String getValue(String input) {
    // Define a regular expression pattern for extracting the subdomain.
    Pattern pattern = Pattern.compile("^(.*)\\.\\.iot\\.\\.us-east-1\\.\\.amazonaws\\.
\\.com");

    // Match the pattern against the input string.
    Matcher matcher = pattern.matcher(input);

    // Check if a match is found.
    if (matcher.find()) {
        // Extract the subdomain from the first capturing group.
        String subdomain = matcher.group(1);
        System.out.println("Extracted subdomain: " + subdomain);
        return subdomain ;
    } else {
        System.out.println("No match found");
    }
    return "" ;
}

public static void searchThings(IotClient iotClient, String queryString){
    SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
        .queryString(queryString)
        .build();

    try {
        // Perform the search and get the result.
        SearchIndexResponse searchIndexResponse =
iotClient.searchIndex(searchIndexRequest);

        // Process the result.
```

```
        if (searchIndexResponse.things().isEmpty()) {
            System.out.println("No things found.");
        } else {
            searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
        }
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

## AWS IoT data SDK for Java 2.x を使用した の例

次のコード例は、AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS IoT data。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- [アクション](#)

### アクション

#### GetThingShadow

次の例は、GetThingShadow を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void getPayload(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        GetThingShadowResponse getThingShadowResponse =
iotPlaneClient.getThingShadow(getThingShadowRequest);

        // Extracting payload from response.
        SdkBytes payload = getThingShadowResponse.payload();
        String payloadString = payload.asUtf8String();
        System.out.println("Received Shadow Data: " + payloadString);

    } catch (IotException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [GetThingShadow](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## UpdateThingShadow

次の例は、UpdateThingShadow を使用する方法を説明しています。



## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void updateShadowThing(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
        \"humidity\":50}}}\"";
        SdkBytes data= SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8 );
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        // Update Thing Shadow.
        iotPlaneClient.updateThingShadow(updateThingShadowRequest);
        System.out.println("Thing Shadow updated successfully.");

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [UpdateThingShadow](#)」の「」を参照してください。 AWS SDK for Java 2.x

## SDK for Java 2.x を使用する Amazon Keyspaces の例

次のコード例は、Amazon Keyspaces AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

## 開始方法

### Hello Amazon Keyspaces

次のコード例は、Amazon Keyspaces の使用を開始する方法を示しています。

#### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [ListKeyspaces](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### CreateKeyspace

次の例は、CreateKeyspace を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        CreateKeyspaceResponse response =
            keyClient.createKeyspace(keyspaceRequest);
        System.out.println("The ARN of the KeySpace is " +
            response.resourceArn());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [CreateKeyspace](#)」の「」を参照してください。  
AWS SDK for Java 2.x

### CreateTable

次の例は、CreateTable を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        // Set the columns.
        ColumnDefinition defTitle = ColumnDefinition.builder()
            .name("title")
            .type("text")
            .build();

        ColumnDefinition defYear = ColumnDefinition.builder()
            .name("year")
            .type("int")
            .build();

        ColumnDefinition defReleaseDate = ColumnDefinition.builder()
            .name("release_date")
            .type("timestamp")
            .build();

        ColumnDefinition defPlot = ColumnDefinition.builder()
            .name("plot")
            .type("text")
            .build();

        List<ColumnDefinition> collist = new ArrayList<>();
        collist.add(defTitle);
        collist.add(defYear);
        collist.add(defReleaseDate);
        collist.add(defPlot);

        // Set the keys.
        PartitionKey yearKey = PartitionKey.builder()
            .name("year")
            .build();
```

```
PartitionKey titleKey = PartitionKey.builder()
    .name("title")
    .build();

List<PartitionKey> keyList = new ArrayList<>();
keyList.add(yearKey);
keyList.add(titleKey);

SchemaDefinition schemaDefinition = SchemaDefinition.builder()
    .partitionKeys(keyList)
    .allColumns(colList)
    .build();

PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
    .status(PointInTimeRecoveryStatus.ENABLED)
    .build();

CreateTableRequest tableRequest = CreateTableRequest.builder()
    .keyspaceName(keySpace)
    .tableName(tableName)
    .schemaDefinition(schemaDefinition)
    .pointInTimeRecovery(timeRecovery)
    .build();

CreateTableResponse response = keyClient.createTable(tableRequest);
System.out.println("The table ARN is " + response.resourceArn());

} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス [CreateTable](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteKeyspace

次の例は、DeleteKeyspace を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [DeleteKeyspace](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteTable

次の例は、DeleteTable を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
    try {
        DeleteTableRequest tableRequest = DeleteTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        keyClient.deleteTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DeleteTable](#)」の「」を参照してください。AWS SDK for Java 2.x

## GetKeyspace

次の例は、GetKeyspace を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [GetKeyspace](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## GetTable

次の例は、GetTable を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
}
}
```

- API の詳細については、「API リファレンス [GetTable](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListKeyspaces

次の例は、ListKeyspaces を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [ListKeyspaces](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListTables

次の例は、ListTables を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
    try {
        ListTablesRequest tablesRequest = ListTablesRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
        listRes.stream()
            .flatMap(r -> r.tables().stream())
            .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
```

```
        " Table name: " + content.tableName()));  
  
    } catch (KeyspacesException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- API の詳細については、「API リファレンス [List Tables](#)」の「」を参照してください。AWS SDK for Java 2.x

## RestoreTable

次の例は、RestoreTable を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);  
    }  
}
```

- API の詳細については、「API リファレンス [RestoreTable](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## UpdateTable

次の例は、UpdateTable を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void updateTable(KeyspacesClient keyClient, String keySpace,  
String tableName) {  
    try {  
        ColumnDefinition def = ColumnDefinition.builder()  
            .name("watched")  
            .type("boolean")  
            .build();  
  
        UpdateTableRequest tableRequest = UpdateTableRequest.builder()  
            .keyspaceName(keySpace)  
            .tableName(tableName)  
            .addColumnns(def)  
            .build();  
  
        keyClient.updateTable(tableRequest);  
  
    } catch (KeyspacesException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

```
    }  
}
```

- API の詳細については、「API リファレンス [UpdateTable](#)」の「」を参照してください。AWS SDK for Java 2.x

## シナリオ

### キースペースとテーブルの使用を開始する

次のコードサンプルは、以下の操作方法を示しています。

- キースペースとテーブルを作成する。テーブルスキーマは映画データを保持し、point-in-time 復旧が有効になっています。
- SIGv4 認証による安全な TLS 接続を使用してキースペースに接続します。
- テーブルに対してクエリを実行します。ムービーデータを追加、取得、更新します。
- テーブルを更新する。視聴したムービーを追跡する列を追加します。
- テーブルを以前の状態に戻し、リソースをクリーンアップします。

### SDK for Java 2.x

#### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**  
 * Before running this Java (v2) code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * Before running this Java code example, you must create a  
 * Java keystore (JKS) file and place it in your project's resources folder.  
 *
```

```
* This file is a secure file format used to hold certificate information for
* Java applications. This is required to make a connection to Amazon Keyspaces.
* For more information, see the following documentation topic:
```

```
*
```

```
* https://docs.aws.amazon.com/keyspaces/latest/devguide/using\_java\_driver.html
```

```
*
```

```
* This Java example performs the following tasks:
```

```
*
```

- \* 1. Create a keyspace.
- \* 2. Check for keyspace existence.
- \* 3. List keyspaces using a paginator.
- \* 4. Create a table with a simple movie data schema and enable point-in-time recovery.
- \* 5. Check for the table to be in an Active state.
- \* 6. List all tables in the keyspace.
- \* 7. Use a Cassandra driver to insert some records into the Movie table.
- \* 8. Get all records from the Movie table.
- \* 9. Get a specific Movie.
- \* 10. Get a UTC timestamp for the current time.
- \* 11. Update the table schema to add a 'watched' Boolean column.
- \* 12. Update an item as watched.
- \* 13. Query for items with watched = True.
- \* 14. Restore the table back to the previous state using the timestamp.
- \* 15. Check for completion of the restore action.
- \* 16. Delete the table.
- \* 17. Confirm that both tables are deleted.
- \* 18. Delete the keyspace.

```
*/
```

```
public class ScenarioKeyspaces {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    /*
     * Usage:
     * fileName - The name of the JSON file that contains movie data. (Get this file
     * from the GitHub repo at resources/sample_file.)
     * keyspaceName - The name of the keyspace to create.
     */
    public static void main(String[] args) throws InterruptedException, IOException
    {
        String fileName = "<Replace with the JSON file that contains movie data>";
        String keyspaceName = "<Replace with the name of the keyspace to create>";
        String titleUpdate = "The Family";
        int yearUpdate = 2013;
```

```
String tableName = "Movie";
String tableNameRestore = "MovieRestore";
Region region = Region.US_EAST_1;
KeyspacesClient keyClient = KeyspacesClient.builder()
    .region(region)
    .build();

DriverConfigLoader loader =
DriverConfigLoader.fromClasspath("application.conf");
CqlSession session = CqlSession.builder()
    .withConfigLoader(loader)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon Keyspaces example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create a keyspace.");
createKeyspace(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
Thread.sleep(5000);
System.out.println("2. Check for keyspace existence.");
checkKeyspaceExistence(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. List keyspaces using a paginator.");
listKeyspacesPaginator(keyClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Create a table with a simple movie data schema and
enable point-in-time recovery.");
createTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Check for the table to be in an Active state.");
Thread.sleep(6000);
checkTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);
```



```
System.out.println(DASHES);
System.out.println("6. List all tables in the keyspace.");
listTables(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Use a Cassandra driver to insert some records into
the Movie table.");
Thread.sleep(6000);
loadData(session, fileName, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Get all records from the Movie table.");
getMovieData(session, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Get a specific Movie.");
getSpecificMovie(session, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get a UTC timestamp for the current time.");
ZonedDateTime utc = ZonedDateTime.now(ZoneOffset.UTC);
System.out.println("DATETIME = " + Date.from(utc.toInstant()));
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Update the table schema to add a watched Boolean
column.");
updateTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Update an item as watched.");
Thread.sleep(10000); // Wait 10 secs for the update.
updateRecord(session, keyspaceName, titleUpdate, yearUpdate);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Query for items with watched = True.");
getWatchedData(session, keyspaceName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Restore the table back to the previous state using
the timestamp.");
        System.out.println("Note that the restore operation can take up to 20
minutes.");
        restoreTable(keyClient, keyspaceName, utc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Check for completion of the restore action.");
        Thread.sleep(5000);
        checkRestoredTable(keyClient, keyspaceName, "MovieRestore");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete both tables.");
        deleteTable(keyClient, keyspaceName, tableName);
        deleteTable(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Confirm that both tables are deleted.");
        checkTableDelete(keyClient, keyspaceName, tableName);
        checkTableDelete(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Delete the keyspace.");
        deleteKeyspace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The scenario has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                .keyspaceName(keyspaceName)
```

```
        .build();

        keyClient.deleteKeyspace(deleteKeyspaceRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkTableDelete(KeyspacesClient keyClient, String
keyspaceName, String tableName)
    throws InterruptedException {
    try {
        String status;
        GetTableResponse response;
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        // Keep looping until table cannot be found and a
ResourceNotFoundException is
        // thrown.
        while (true) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println(". The table status is " + status);
            Thread.sleep(500);
        }

    } catch (ResourceNotFoundException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println("The table is deleted");
}

public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
    try {
        DeleteTableRequest tableRequest = DeleteTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();
```

```
        keyClient.deleteTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkRestoredTable(KeyspacesClient keyClient, String
keyspaceName, String tableName)
    throws InterruptedException {
    try {
        boolean tableStatus = false;
        String status;
        GetTableResponse response = null;
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        while (!tableStatus) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println("The table status is " + status);

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true;
            }
            Thread.sleep(500);
        }

        List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
        for (ColumnDefinition def : cols) {
            System.out.println("The column name is " + def.name());
            System.out.println("The column type is " + def.type());
        }

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```

    public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
    ZonedDateTime utc) {
        try {
            Instant myTime = utc.toInstant();
            RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
                .restoreTimestamp(myTime)
                .sourceTableName("Movie")
                .targetKeyspaceName(keyspaceName)
                .targetTableName("MovieRestore")
                .sourceKeyspaceName(keyspaceName)
                .build();

            RestoreTableResponse response =
            keyClient.restoreTable(restoreTableRequest);
            System.out.println("The ARN of the restored table is " +
            response.restoredTableARN());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getWatchedData(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session
            .execute("SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE
watched = true ALLOW FILTERING;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    public static void updateRecord(CqlSession session, String keySpace, String
titleUpdate, int yearUpdate) {
        String sqlStatement = "UPDATE \"" + keySpace
            + "\".\"Movie\" SET watched=true WHERE title = :k0 AND year = :k1;";
        BatchStatementBuilder builder =
        BatchStatement.builder(DefaultBatchType.UNLOGGED);
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
        PreparedStatement preparedStatement = session.prepare(sqlStatement);
        builder.addStatement(preparedStatement.boundStatementBuilder()
            .setString("k0", titleUpdate)

```

```
        .setInt("k1", yearUpdate)
        .build());

    BatchStatement batchStatement = builder.build();
    session.execute(batchStatement);
}

public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        ColumnDefinition def = ColumnDefinition.builder()
            .name("watched")
            .type("boolean")
            .build();

        UpdateTableRequest tableRequest = UpdateTableRequest.builder()
            .keyspaceName(keySpace)
            .tableName(tableName)
            .addColumnns(def)
            .build();

        keyClient.updateTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getSpecificMovie(CqlSession session, String keyspaceName) {
    ResultSet resultSet = session.execute(
        "SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE title = 'The
Family' ALLOW FILTERING ;");
    resultSet.forEach(item -> {
        System.out.println("The Movie title is " + item.getString("title"));
        System.out.println("The Movie year is " + item.getInt("year"));
        System.out.println("The plot is " + item.getString("plot"));
    });
}

// Get records from the Movie table.
public static void getMovieData(CqlSession session, String keyspaceName) {
    ResultSet resultSet = session.execute("SELECT * FROM \"" + keyspaceName +
"\".\"Movie\";");
}
```

```
resultSet.forEach(item -> {
    System.out.println("The Movie title is " + item.getString("title"));
    System.out.println("The Movie year is " + item.getInt("year"));
    System.out.println("The plot is " + item.getString("plot"));
});
}

// Load data into the table.
public static void loadData(CqlSession session, String fileName, String
keySpace) throws IOException {
    String sqlStatement = "INSERT INTO \"" + keySpace + "\".\"Movie\" (title,
year, plot) values (:k0, :k1, :k2)";
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {

        // Add 20 movies to the table.
        if (t == 20)
            break;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String plot = currentNode.path("info").path("plot").toString();

        // Insert the data into the Amazon Keyspaces table.
        BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
        PreparedStatement preparedStatement = session.prepare(sqlStatement);
        builder.addStatement(preparedStatement.boundStatementBuilder()
            .setString("k0", title)
            .setInt("k1", year)
            .setString("k2", plot)
            .build());

        BatchStatement batchStatement = builder.build();
        session.execute(batchStatement);
        t++;
    }
}
```

```
        System.out.println("You have added " + t + " records successfully!");
    }

    public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
        try {
            ListTablesRequest tablesRequest = ListTablesRequest.builder()
                .keyspaceName(keyspaceName)
                .build();

            ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
            listRes.stream()
                .flatMap(r -> r.tables().stream())
                .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                    " Table name: " + content.tableName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
        throws InterruptedException {
        try {
            boolean tableStatus = false;
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                .keyspaceName(keyspaceName)
                .tableName(tableName)
                .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is " + status);

                if (status.compareTo("ACTIVE") == 0) {
                    tableStatus = true;
                }
            }
        }
    }
}
```



```
        Thread.sleep(500);
    }

    List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
    for (ColumnDefinition def : cols) {
        System.out.println("The column name is " + def.name());
        System.out.println("The column type is " + def.type());
    }

} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        // Set the columns.
        ColumnDefinition defTitle = ColumnDefinition.builder()
            .name("title")
            .type("text")
            .build();

        ColumnDefinition defYear = ColumnDefinition.builder()
            .name("year")
            .type("int")
            .build();

        ColumnDefinition defReleaseDate = ColumnDefinition.builder()
            .name("release_date")
            .type("timestamp")
            .build();

        ColumnDefinition defPlot = ColumnDefinition.builder()
            .name("plot")
            .type("text")
            .build();

        List<ColumnDefinition> collist = new ArrayList<>();
        collist.add(defTitle);
        collist.add(defYear);
        collist.add(defReleaseDate);
        collist.add(defPlot);
    }
}
```

```
// Set the keys.
PartitionKey yearKey = PartitionKey.builder()
    .name("year")
    .build();

PartitionKey titleKey = PartitionKey.builder()
    .name("title")
    .build();

List<PartitionKey> keyList = new ArrayList<>();
keyList.add(yearKey);
keyList.add(titleKey);

SchemaDefinition schemaDefinition = SchemaDefinition.builder()
    .partitionKeys(keyList)
    .allColumns(colList)
    .build();

PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
    .status(PointInTimeRecoveryStatus.ENABLED)
    .build();

CreateTableRequest tableRequest = CreateTableRequest.builder()
    .keyspaceName(keySpace)
    .tableName(tableName)
    .schemaDefinition(schemaDefinition)
    .pointInTimeRecovery(timeRecovery)
    .build();

CreateTableResponse response = keyClient.createTable(tableRequest);
System.out.println("The table ARN is " + response.resourceArn());

} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
    try {
        ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
            .maxResults(10)
            .build();
```

```
        ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
        listRes.stream()
            .flatMap(r -> r.keyspaces().stream())
            .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
        String name = response.keyspaceName();
        System.out.println("The " + name + " KeySpace is ready");

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
        System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

    } catch (KeyspacesException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [CreateKeyspace](#)
  - [CreateTable](#)
  - [DeleteKeyspace](#)
  - [DeleteTable](#)
  - [GetKeyspace](#)
  - [GetTable](#)
  - [ListKeyspaces](#)
  - [ListTables](#)
  - [RestoreTable](#)
  - [UpdateTable](#)

## SDK for Java 2.x を使用した Kinesis の例

次のコード例は、Kinesis AWS SDK for Java 2.x でを使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### トピック

- [アクション](#)


- [サーバーレスサンプル](#)

## アクション

### CreateStream

次の例は、CreateStream を使用する方法を説明しています。

SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [CreateStream](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteStream

次の例は、DeleteStream を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [DeleteStream](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## GetRecords

次の例は、GetRecords を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.SdkBytes;
```



```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.Shard;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorRequest;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorResponse;
import software.amazon.awssdk.services.kinesis.model.Record;
import software.amazon.awssdk.services.kinesis.model.GetRecordsRequest;
import software.amazon.awssdk.services.kinesis.model.GetRecordsResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetRecords {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <streamName>

            Where:
                streamName - The Amazon Kinesis data stream to read from (for
example, StockTradeStream).
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
            .region(region)
            .build();
    }
}
```

```
        getStockTrades(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void getStockTrades(KinesisClient kinesisClient, String
streamName) {
        String shardIterator;
        String lastShardId = null;
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
            .streamName(streamName)
            .build();

        List<Shard> shards = new ArrayList<>();
        DescribeStreamResponse streamRes;
        do {
            streamRes = kinesisClient.describeStream(describeStreamRequest);
            shards.addAll(streamRes.streamDescription().shards());

            if (shards.size() > 0) {
                lastShardId = shards.get(shards.size() - 1).shardId();
            }
        } while (streamRes.streamDescription().hasMoreShards());

        GetShardIteratorRequest itReq = GetShardIteratorRequest.builder()
            .streamName(streamName)
            .shardIteratorType("TRIM_HORIZON")
            .shardId(lastShardId)
            .build();

        GetShardIteratorResponse shardIteratorResult =
kinesisClient.getShardIterator(itReq);
        shardIterator = shardIteratorResult.shardIterator();

        // Continuously read data records from shard.
        List<Record> records;

        // Create new GetRecordsRequest with existing shardIterator.
        // Set maximum records to return to 1000.
        GetRecordsRequest recordsRequest = GetRecordsRequest.builder()
            .shardIterator(shardIterator)
            .limit(1000)
            .build();
```

```
GetRecordsResponse result = kinesisClient.getRecords(recordsRequest);

// Put result into record list. Result may be empty.
records = result.records();

// Print records
for (Record record : records) {
    SdkBytes byteBuffer = record.data();
    System.out.printf("Seq No: %s - %s%n", record.sequenceNumber(), new
String(byteBuffer.asByteArray()));
}
}
```

- API の詳細については、「API リファレンス [GetRecords](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutRecord

次の例は、PutRecord を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「API リファレンス [PutRecord](#)」の「」を参照してください。AWS SDK for Java 2.x

## サーバーレスサンプル

### Kinesis トリガーから Lambda 関数を呼び出す

次のコード例では、Kinesis ストリームからレコードを受信することによってトリガーされるイベントを受け取る、Lambda 関数の実装方法を示しています。この関数は Kinesis ペイロードを取得し、それを Base64 からデコードして、そのレコードの内容をログ記録します。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Java を使用した Lambda での Kinesis イベントの消費。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+" records");
        return null;
    }
}
```

```
}
```

## Kinesis トリガーを使用した Lambda 関数でのバッチアイテムの失敗のレポート

以下のコード例では、Kinesis ストリームからイベントを受け取る Lambda 関数のための、部分的なバッチレスポンスの実装方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [サーバーレスサンプル](#) リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

## Java を使用した Lambda での Kinesis バッチアイテム失敗のレポート。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
ArrayList<>();
        String curRecordSequenceNumber = "";
```



```
        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
immediately.
                Lambda will immediately begin to retry processing from this
failed item onwards. */
                batchItemFailures.add(new
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

        return new StreamsEventResponse(batchItemFailures);
    }
}
```

## AWS KMS SDK for Java 2.x を使用した の例

次のコード例は、AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS KMS。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### 開始方法

## Hello KMS キー

次のコード例は、KMS キーの使用を開始する方法を示しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.paginators.ListKeysIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        listAllKeys(kmsClient);
        kmsClient.close();
    }

    public static void listAllKeys(KmsClient kmsClient) {
        try {
            ListKeysRequest listKeysRequest = ListKeysRequest.builder()
                .limit(15)
                .build();
        }
    }
}
```

```
ListKeysIterable keysResponse =
kmsClient.listKeysPaginator(listKeysRequest);
keysResponse.stream()
    .flatMap(r -> r.keys().stream())
    .forEach(key -> System.out
        .println(" The key ARN is: " + key.keyArn() + ". The key Id is:
" + key.keyId()));

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス[listKeysPaginator](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### CreateAlias

次の例は、CreateAlias を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,
String aliasName) {
    try {
```

```
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        kmsClient.createAlias(aliasRequest);
        System.out.println(aliasName + " was successfully created.");

    } catch (ResourceExistsException e) {
        System.err.println("Alias already exists: " + e.getMessage());
        System.err.println("Moving on...");
    } catch (Exception e) {
        System.err.println("An unexpected error occurred: " + e.getMessage());
        System.err.println("Moving on...");
    }
}
```

- API の詳細については、「API リファレンス [CreateAlias](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateGrant

次の例は、CreateGrant を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String grantKey(KmsClient kmsClient, String keyId, String
granteePrincipal) {
    try {
        // Add the desired KMS Grant permissions.
        List<GrantOperation> grantPermissions = new ArrayList<>();
        grantPermissions.add(GrantOperation.ENCRYPT);
        grantPermissions.add(GrantOperation.DECRYPT);
        grantPermissions.add(GrantOperation.DESCRIBE_KEY);
```

```
        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CreateGrantResponse response = kmsClient.createGrant(grantRequest);
        return response.grantId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- API の詳細については、「API リファレンス [CreateGrant](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateKey

次の例は、CreateKey を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String createKey(KmsClient kmsClient, String keyDesc) {
    try {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
            .keyUsage("ENCRYPT_DECRYPT")
            .build();
```

```
        CreateKeyResponse result = kmsClient.createKey(keyRequest);
        System.out.println("Symmetric key with ARN [" +
result.keyMetadata().arn() + "] has been created.");
        return result.keyMetadata().keyId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- APIの詳細については、「APIリファレンス[CreateKey](#)」の「」を参照してください。AWS SDK for Java 2.x

## Decrypt

次の例は、Decrypt を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String decryptData(KmsClient kmsClient, SdkBytes encryptedData,
String keyId) {
    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
        return decryptResponse.plaintext().asString(StandardCharsets.UTF_8);

    } catch (KmsException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[Decrypt](#)」を参照してください。

## DeleteAlias

次の例は、DeleteAlias を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void deleteSpecificAlias(KmsClient kmsClient, String aliasName) {
    try {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        kmsClient.deleteAlias(deleteAliasRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DeleteAlias](#)」の「」を参照してください。 AWS SDK for Java 2.x

## DescribeKey

次の例は、DescribeKey を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static boolean isKeyEnabled(KmsClient kmsClient, String keyId) {
    try {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
        KeyState keyState = response.keyMetadata().keyState();
        if (keyState == KeyState.ENABLED) {
            System.out.println("The key is enabled.");
            return true;
        } else {
            System.out.println("The key is not enabled. Key state: " +
keyState);
        }

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return false;
    }
}
```

- API の詳細については、「API リファレンス [DescribeKey](#)」の「」を参照してください。 AWS SDK for Java 2.x



## DisableKey

次の例は、DisableKey を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void disableKey(KmsClient kmsClient, String keyId) {
    try {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.disableKey(keyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DisableKey](#)」の「」を参照してください。 AWS SDK for Java 2.x

## EnableKey

次の例は、EnableKey を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Enable the KMS key.
public static void enableKey(KmsClient kmsClient, String keyId) {
    try {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.enableKey(enableKeyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [EnableKey](#)」の「」を参照してください。AWS SDK for Java 2.x

## Encrypt

次の例は、Encrypt を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static SdkBytes encryptData(KmsClient kmsClient, String keyId, String
text) {
    try {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        EncryptResponse response = kmsClient.encrypt(encryptRequest);
    }
}
```

```
String algorithm = response.encryptionAlgorithm().toString();
System.out.println("The string was encrypted with algorithm " +
algorithm + ".");

// Get the encrypted data.
SdkBytes encryptedData = response.ciphertextBlob();
return encryptedData;

} catch (KmsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return null;
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[Encrypt](#)」を参照してください。

## ListAliases

次の例は、ListAliases を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void listAllAliases(KmsClient kmsClient) {
    try {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesIterable aliasesResponse =
            kmsClient.listAliasesPaginator(aliasesRequest);
        aliasesResponse.stream()
            .flatMap(r -> r.aliases().stream())
```

```
        .forEach(alias -> System.out
            .println("The alias name is: " + alias.aliasName()));
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [ListAliases](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListGrants

次の例は、ListGrants を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void displayGrantIds(KmsClient kmsClient, String keyId) {
    try {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();

        ListGrantsIterable response =
            kmsClient.listGrantsPaginator(grantsRequest);
        response.stream()
            .flatMap(r -> r.grants().stream())
            .forEach(grant -> {
                System.out.println("The grant Id is : " + grant.grantId());
                List<GrantOperation> ops = grant.operations();
                for (GrantOperation op : ops) {
                    System.out.println(op.name());
                }
            });
    }
}
```

```
        }
    });

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [ListGrants](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListKeyPolicies

次の例は、ListKeyPolicies を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void getKeyPolicy(KmsClient kmsClient, String keyId, String
policyName) {
    try {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        GetKeyPolicyResponse response = kmsClient.getKeyPolicy(policyRequest);
        System.out.println("The response is "+response.policy());
    } catch (KmsException e) {
        if (e.getMessage().contains("No such policy exists")) {
            System.out.println("The policy cannot be found. Error message: " +
e.getMessage());
        } else {
            throw e;
        }
    }
}
```

```
    }  
  }  
}
```

- API の詳細については、「API リファレンス [ListKeyPolicies](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListKeys

次の例は、ListKeys を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.kms.KmsClient;  
import software.amazon.awssdk.services.kms.model.ListKeysRequest;  
import software.amazon.awssdk.services.kms.model.KmsException;  
import software.amazon.awssdk.services.kms.paginators.ListKeysIterable;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class HelloKMS {  
    public static void main(String[] args) {  
        Region region = Region.US_WEST_2;  
        KmsClient kmsClient = KmsClient.builder()  
            .region(region)  
            .build();  
    }  
}
```

```
        listAllKeys(kmsClient);
        kmsClient.close();
    }

    public static void listAllKeys(KmsClient kmsClient) {
        try {
            ListKeysRequest listKeysRequest = ListKeysRequest.builder()
                .limit(15)
                .build();

            ListKeysIterable keysResponse =
kmsClient.listKeysPaginator(listKeysRequest);
            keysResponse.stream()
                .flatMap(r -> r.keys().stream())
                .forEach(key -> System.out
                    .println(" The key ARN is: " + key.keyArn() + ". The key Id is:
" + key.keyId()));

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- APIの詳細については、「APIリファレンス[ListKeys](#)」の「」を参照してください。AWS SDK for Java 2.x

## RevokeGrant

次の例は、RevokeGrant を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void revokeKeyGrant(KmsClient kmsClient, String keyId, String
grantId) {
    try {
        RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
            .keyId(keyId)
            .grantId(grantId)
            .build();

        kmsClient.revokeGrant(grantRequest);
        System.out.println("Grant ID: [" + grantId + "] was successfully
revoke!");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [RevokeGrant](#)」の「」を参照してください。AWS SDK for Java 2.x

## ScheduleKeyDeletion

次の例は、ScheduleKeyDeletion を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void deleteKey(KmsClient kmsClient, String keyId) {
    try {
        ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
            .pendingWindowInDays(7)
            .build();
```



```
kmsClient.scheduleKeyDeletion(deletionRequest);
System.out.println("The key will be deleted in 7 days.");

} catch (KmsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス [ScheduleKeyDeletion](#)」の「」を参照してください。AWS SDK for Java 2.x

## Sign

次の例は、Sign を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void signVerifyData(KmsClient kmsClient) {
    String signMessage = "Here is the message that will be digitally signed";

    // Create an AWS KMS key used to digitally sign data.
    CreateKeyRequest request = CreateKeyRequest.builder()
        .keySpec(KeySpec.RSA_2048) // Specify key spec
        .keyUsage(KeyUsageType.SIGN_VERIFY) // Specify key usage
        .origin(OriginType.AWS_KMS) // Specify key origin
        .build();

    CreateKeyResponse response = kmsClient.createKey(request);
    String keyId2 = response.keyMetadata().keyId();
    System.out.println("Created KMS key with ID: " + keyId2);

    SdkBytes bytes = SdkBytes.fromString(signMessage, Charset.defaultCharset());
}
```

```
SignRequest signRequest = SignRequest.builder()
    .keyId(keyId2)
    .message(bytes)
    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
    .build();

SignResponse signResponse = kmsClient.sign(signRequest);
byte[] signedBytes = signResponse.signature().asByteArray();

// Verify the digital signature.
VerifyRequest verifyRequest = VerifyRequest.builder()
    .keyId(keyId2)

    .message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))
    .signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))
    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
    .build();

VerifyResponse verifyResponse = kmsClient.verify(verifyRequest);
System.out.println("Signature verification result: " +
verifyResponse.signatureValid());
}
```

- APIの詳細については、「API リファレンスで[サインイン](#)」を参照してください。AWS SDK for Java 2.x

## TagResource

次の例は、TagResource を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void tagKMSKey(KmsClient kmsClient, String keyId) {
    try {
```

```
Tag tag = Tag.builder()
    .tagKey("Environment")
    .tagValue("Production")
    .build();

TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
    .keyId(keyId)
    .tags(tag)
    .build();

kmsClient.tagResource(tagResourceRequest);
System.out.println("The key has been tagged.");

} catch (KmsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス [TagResource](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## シナリオ

KMS キーのコアオペレーションについて説明します。

次のコードサンプルは、以下の操作方法を示しています。

- KMS キーを作成します。
- アカウントの KMS キーを一覧表示し、詳細を取得します。
- KMS キーを有効および無効にします。
- クライアント側の暗号化に使用できる対称データキーを生成します。
- データのデジタル署名に使用される非対称キーを生成します。
- タグキー。
- KMS キーを削除します。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.AliasListEntry;
import software.amazon.awssdk.services.kms.model.AlreadyExistsException;
import software.amazon.awssdk.services.kms.model.CreateAliasRequest;
import software.amazon.awssdk.services.kms.model.CreateGrantRequest;
import software.amazon.awssdk.services.kms.model.CreateGrantResponse;
import software.amazon.awssdk.services.kms.model.CreateKeyRequest;
import software.amazon.awssdk.services.kms.model.CreateKeyResponse;
import software.amazon.awssdk.services.kms.model.CustomerMasterKeySpec;
import software.amazon.awssdk.services.kms.model.DecryptRequest;
import software.amazon.awssdk.services.kms.model.DecryptResponse;
import software.amazon.awssdk.services.kms.model.DeleteAliasRequest;
import software.amazon.awssdk.services.kms.model.DescribeKeyRequest;
import software.amazon.awssdk.services.kms.model.DescribeKeyResponse;
import software.amazon.awssdk.services.kms.model.DisableKeyRequest;
import software.amazon.awssdk.services.kms.model.EnableKeyRequest;
import software.amazon.awssdk.services.kms.model.EnableKeyRotationRequest;
import software.amazon.awssdk.services.kms.model.EncryptRequest;
import software.amazon.awssdk.services.kms.model.EncryptResponse;
import software.amazon.awssdk.services.kms.model.GetKeyPolicyRequest;
import software.amazon.awssdk.services.kms.model.GetKeyPolicyResponse;
import software.amazon.awssdk.services.kms.model.GrantOperation;
import software.amazon.awssdk.services.kms.model.KeySpec;
import software.amazon.awssdk.services.kms.model.KeyState;
import software.amazon.awssdk.services.kms.model.KeyUsageType;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.LimitExceededException;
import software.amazon.awssdk.services.kms.model.ListAliasesRequest;
import software.amazon.awssdk.services.kms.model.ListGrantsRequest;
import software.amazon.awssdk.services.kms.model.ListKeyPoliciesRequest;
import software.amazon.awssdk.services.kms.model.ListKeyPoliciesResponse;
import software.amazon.awssdk.services.kms.model.OriginType;
```

```
import software.amazon.awssdk.services.kms.model.PutKeyPolicyRequest;
import software.amazon.awssdk.services.kms.model.RevokeGrantRequest;
import software.amazon.awssdk.services.kms.model.ScheduleKeyDeletionRequest;
import software.amazon.awssdk.services.kms.model.SignRequest;
import software.amazon.awssdk.services.kms.model.SignResponse;
import software.amazon.awssdk.services.kms.model.SigningAlgorithmSpec;
import software.amazon.awssdk.services.kms.model.Tag;
import software.amazon.awssdk.services.kms.model.TagResourceRequest;
import software.amazon.awssdk.services.kms.model.VerifyRequest;
import software.amazon.awssdk.services.kms.model.VerifyResponse;
import software.amazon.awssdk.services.kms.paginators.ListAliasesIterable;
import software.amazon.awssdk.services.kms.paginators.ListGrantsIterable;
import software.amazon.awssdk.services.secretsmanager.model.ResourceExistsException;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.GetCallerIdentityResponse;
import java.nio.ByteBuffer;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.util.List;
import java.util.ArrayList;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class KMSScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String accountId = getAccountId();

    public static void main(String[] args) {
        final String usage = ""
            Usage: <granteePrincipal>

        Where:
            granteePrincipal - The principal (user, service account, or
            group) to whom the grant or permission is being given.
        """;
    }
}
```

```
    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }
    String granteePrincipal = args[0];
    String policyName = "default";

    Scanner scanner = new Scanner(System.in);
    String keyDesc = "Created by the AWS KMS API";

    Region region = Region.US_WEST_2;
    KmsClient kmsClient = KmsClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("""
        Welcome to the AWS Key Management SDK Getting Started scenario.

        This program demonstrates how to interact with AWS Key Management using
the AWS SDK for Java (v2).
        The AWS Key Management Service (KMS) is a secure and highly available
service that allows you to create
        and manage AWS KMS keys and control their use across a wide range of AWS
services and applications.
        KMS provides a centralized and unified approach to managing encryption
keys, making it easier to meet your
        data protection and regulatory compliance requirements.

        This Getting Started scenario creates two key types. A symmetric
encryption key is used to encrypt and decrypt data,
        and an asymmetric key used to digitally sign data.
        Let's get started...
        """);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("1. Create a symmetric KMS key\n");
    System.out.println("First, the program will creates a symmetric KMS key that
you can used to encrypt and decrypt data.");
    waitForInputToContinue(scanner);
    String targetKeyId = createKey(kmsClient, keyDesc);
    waitForInputToContinue(scanner);
```

```

System.out.println(DASHES);
System.out.println("""
    2. Enable a KMS key

```

By default, when the SDK creates an AWS key it is enabled. The next bit of code checks to determine if the key is enabled. If it is not enabled, the code enables it.

```

    """);
waitForInputToContinue(scanner);
boolean isEnabled = isKeyEnabled(kmsClient, targetKeyId);
if (!isEnabled)
    enableKey(kmsClient, targetKeyId);
waitForInputToContinue(scanner);

```

```

System.out.println(DASHES);
System.out.println("3. Encrypt data using the symmetric KMS key");
String plaintext = "Hello, AWS KMS!";
System.out.printf("""

```

One of the main uses of symmetric keys is to encrypt and decrypt data.

Next, the code encrypts the string '%s' with the SYMMETRIC\_DEFAULT encryption algorithm.

```

    %n""", plaintext);
waitForInputToContinue(scanner);
SdkBytes ciphertext = encryptData(kmsClient, targetKeyId, plaintext);
waitForInputToContinue(scanner);

```

```

System.out.println(DASHES);
System.out.println("4. Create an alias");
System.out.println("""

```

Enter an alias name for the key. The name should be prefixed with 'alias/'.

For example, 'alias/myFirstKey'.

```

    """);

```

```

String aliasName = scanner.nextLine();
String fullAliasName = aliasName.isEmpty() ? "alias/dev-encryption-key" :
aliasName;
createCustomAlias(kmsClient, targetKeyId, fullAliasName);
waitForInputToContinue(scanner);

```

```
System.out.println(DASHES);
System.out.println("5. List all of your aliases");
waitForInputToContinue(scanner);
listAllAliases(kmsClient);
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("6. Enable automatic rotation of the KMS key");
System.out.println("""
```

By default, when the SDK enables automatic rotation of a KMS key, KMS rotates the key material of the KMS key one year (approximately 365 days) from the enable date and every year thereafter.

```
""");
waitForInputToContinue(scanner);
enableKeyRotation(kmsClient, targetKeyId);
waitForInputToContinue(scanner);
```

```
System.out.println(DASHES);
System.out.println("""
    7. Create a grant
```

A grant is a policy instrument that allows Amazon Web Services principals to use KMS keys.

It also can allow them to view a KMS key (DescribeKey) and create and manage grants.

When authorizing access to a KMS key, grants are considered along with key policies and IAM policies.

```
""");

waitForInputToContinue(scanner);
String grantId = grantKey(kmsClient, targetKeyId, granteePrincipal);
System.out.println("The code granted principal with ARN [" +
granteePrincipal + "] ");
System.out.println("use of the symmetric key. The grant ID is [" + grantId +
" ]");
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("8. List grants for the KMS key");
waitForInputToContinue(scanner);
displayGrantIds(kmsClient, targetKeyId);
waitForInputToContinue(scanner);
```



```
System.out.println(DASHES);
System.out.println("9. Revoke the grant");
waitForInputToContinue(scanner);
revokeKeyGrant(kmsClient, targetKeyId, grantId);
waitForInputToContinue(scanner);
```

```
System.out.println(DASHES);
System.out.println("10. Decrypt the data\n");
System.out.println("""
```

Lets decrypt the data that was encrypted in an early step.  
The code uses the same key to decrypt the string that we encrypted earlier in the program.

```
""");
waitForInputToContinue(scanner);
String decryptText = decryptData(kmsClient, ciphertext, targetKeyId);
System.out.println("Decrypted text is: " + decryptText);
waitForInputToContinue(scanner);
```

```
System.out.println(DASHES);
System.out.println("11. Replace a key policy\n");
System.out.println("""
```

A key policy is a resource policy for a KMS key. Key policies are the primary way to control access to KMS keys. Every KMS key must have exactly one key policy. The statements in the key policy determine who has permission to use the KMS key and how they can use it.

You can also use IAM policies and grants to control access to the KMS key, but every KMS key must have a key policy.

By default, when you create a key by using the SDK, a policy is created that gives the AWS account that owns the KMS key full access to the KMS key.

Let's try to replace the automatically created policy with the following policy.

```
"Version": "2012-10-17",
"Statement": [{
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::0000000000:root"},
  "Action": "kms:*",
```

```
        "Resource": "*"
    }]
    """);

    waitForInputToContinue(scanner);
    boolean polAdded = replacePolicy(kmsClient, targetKeyId, policyName);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("12. Get the key policy\n");
    System.out.println("The next bit of code that runs gets the key policy to
make sure it exists.");
    waitForInputToContinue(scanner);
    getKeyPolicy(kmsClient, targetKeyId, policyName);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("13. Create an asymmetric KMS key and sign your data\n");
    System.out.println("""
        Signing your data with an AWS key can provide several benefits that make
it an attractive option
        for your data signing needs. By using an AWS KMS key, you can leverage
the
        security controls and compliance features provided by AWS,
        which can help you meet various regulatory requirements and enhance the
overall security posture
        of your organization.
    """);
    waitForInputToContinue(scanner);
    signVerifyData(kmsClient);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("14. Tag your symmetric KMS Key\n");
    System.out.println("""
        By using tags, you can improve the overall management, security, and
governance of your
        KMS keys, making it easier to organize, track, and control access to
your encrypted data within
        your AWS environment
    """);
    waitForInputToContinue(scanner);
    tagKMSKey(kmsClient, targetKeyId);
    waitForInputToContinue(scanner);
```

```
System.out.println(DASHES);
System.out.println("15. Schedule the deletion of the KMS key\n");
System.out.println("""
    By default, KMS applies a waiting period of 30 days,
    but you can specify a waiting period of 7-30 days. When this operation
is successful,
    the key state of the KMS key changes to PendingDeletion and the key
can't be used in any
    cryptographic operations. It remains in this state for the duration of
the waiting period.

    Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
    all data that was encrypted under the KMS key is unrecoverable.\s
""");
System.out.println("Would you like to delete the Key Management resources?
(y/n)");
String delAns = scanner.nextLine().trim();
if (delAns.equalsIgnoreCase("y")) {
    System.out.println("You selected to delete the AWS KMS resources.");
    waitForInputToContinue(scanner);
    deleteSpecificAlias(kmsClient, fullAliasName);
    disableKey(kmsClient, targetKeyId);
    deleteKey(kmsClient, targetKeyId);

} else {
    System.out.println("The Key Management resources will not be deleted");
}

System.out.println(DASHES);
System.out.println("This concludes the AWS Key Management SDK Getting
Started scenario");
System.out.println(DASHES);
}
public static void listAllAliases(KmsClient kmsClient) {
    try {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesIterable aliasesResponse =
kmsClient.listAliasesPaginator(aliasesRequest);
        aliasesResponse.stream()
```

```
        .flatMap(r -> r.aliases().stream())
        .forEach(alias -> System.out
            .println("The alias name is: " + alias.aliasName()));

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void disableKey(KmsClient kmsClient, String keyId) {
    try {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.disableKey(keyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void signVerifyData(KmsClient kmsClient) {
    String signMessage = "Here is the message that will be digitally signed";

    // Create an AWS KMS key used to digitally sign data.
    CreateKeyRequest request = CreateKeyRequest.builder()
        .keySpec(KeySpec.RSA_2048) // Specify key spec
        .keyUsage(KeyUsageType.SIGN_VERIFY) // Specify key usage
        .origin(OriginType.AWS_KMS) // Specify key origin
        .build();

    CreateKeyResponse response = kmsClient.createKey(request);
    String keyId2 = response.keyMetadata().keyId();
    System.out.println("Created KMS key with ID: " + keyId2);

    SdkBytes bytes = SdkBytes.fromString(signMessage, Charset.defaultCharset());
    SignRequest signRequest = SignRequest.builder()
        .keyId(keyId2)
        .message(bytes)
        .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
        .build();
}
```

```
SignResponse signResponse = kmsClient.sign(signRequest);
byte[] signedBytes = signResponse.signature().asByteArray();

// Verify the digital signature.
VerifyRequest verifyRequest = VerifyRequest.builder()
    .keyId(keyId2)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset()))
    .signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))
    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
    .build());

VerifyResponse verifyResponse = kmsClient.verify(verifyRequest);
System.out.println("Signature verification result: " +
verifyResponse.signatureValid());
}

public static void tagKMSKey(KmsClient kmsClient, String keyId) {
    try {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        kmsClient.tagResource(tagResourceRequest);
        System.out.println("The key has been tagged.");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getKeyPolicy(KmsClient kmsClient, String keyId, String
policyName) {
    try {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
```

```
        .policyName(policyName)
        .build();

        GetKeyPolicyResponse response = kmsClient.getKeyPolicy(policyRequest);
        System.out.println("The response is "+response.policy());
    } catch (KmsException e) {
        if (e.getMessage().contains("No such policy exists")) {
            System.out.println("The policy cannot be found. Error message: " +
e.getMessage());
        } else {
            throw e;
        }
    }
}

public static boolean replacePolicy(KmsClient kmsClient, String keyId, String
policyName) {
    // Change the principle in the below JSON.
    String policy = ""
    {
        "Version": "2012-10-17",
        "Statement": [{
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::%s:root"},
            "Action": "kms:*",
            "Resource": "*"
        }]
    }
    """.formatted(accountId);

    try {
        PutKeyPolicyRequest keyPolicyRequest = PutKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .policy(policy)
            .build();
        kmsClient.putKeyPolicy(keyPolicyRequest);
        System.out.println("The key policy has been replaced.");
    } catch (LimitExceededException e) {
        System.out.println("Policy limit reached. Unable to create the
policy.");
        return false;
    } catch (AlreadyExistsException e) {
```

```
        System.out.println("Only one policy per key is supported. Unable to
create the policy.");
        return false;
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    return true;
}

public static boolean doesKeyHavePolicy(KmsClient kmsClient, String keyId,
String policyName){
    ListKeyPoliciesRequest policiesRequest = ListKeyPoliciesRequest.builder()
        .keyId(keyId)
        .build();

    boolean hasPolicy = false;
    ListKeyPoliciesResponse response =
kmsClient.listKeyPolicies(policiesRequest);
    List<String>policyNames = response.policyNames();
    for (String pol : policyNames) {
        hasPolicy = true;
    }
    return hasPolicy;
}

public static void deleteKey(KmsClient kmsClient, String keyId) {
    try {
        ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
            .pendingWindowInDays(7)
            .build();

        kmsClient.scheduleKeyDeletion(deletionRequest);
        System.out.println("The key will be deleted in 7 days.");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
public static void deleteSpecificAlias(KmsClient kmsClient, String aliasName) {
    try {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        kmsClient.deleteAlias(deleteAliasRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static boolean isKeyEnabled(KmsClient kmsClient, String keyId) {
    try {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
        KeyState keyState = response.keyMetadata().keyState();
        if (keyState == KeyState.ENABLED) {
            System.out.println("The key is enabled.");
            return true;
        } else {
            System.out.println("The key is not enabled. Key state: " +
keyState);
        }

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return false;
}

public static String decryptData(KmsClient kmsClient, SdkBytes encryptedData,
String keyId) {
    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();
```



```
        DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
        return decryptResponse.plaintext().asString(StandardCharsets.UTF_8);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void revokeKeyGrant(KmsClient kmsClient, String keyId, String
grantId) {
    try {
        RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
            .keyId(keyId)
            .grantId(grantId)
            .build();

        kmsClient.revokeGrant(grantRequest);
        System.out.println("Grant ID: [" + grantId + "] was successfully
revoke!");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void displayGrantIds(KmsClient kmsClient, String keyId) {
    try {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();

        ListGrantsIterable response =
kmsClient.listGrantsPaginator(grantsRequest);
        response.stream()
            .flatMap(r -> r.grants().stream())
            .forEach(grant -> {
                System.out.println("The grant Id is : " + grant.grantId());
                List<GrantOperation> ops = grant.operations();
                for (GrantOperation op : ops) {
```

```
        System.out.println(op.name());
    }
    });

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String grantKey(KmsClient kmsClient, String keyId, String
granteePrincipal) {
    try {
        // Add the desired KMS Grant permissions.
        List<GrantOperation> grantPermissions = new ArrayList<>();
        grantPermissions.add(GrantOperation.ENCRYPT);
        grantPermissions.add(GrantOperation.DECRYPT);
        grantPermissions.add(GrantOperation.DESCRIBE_KEY);

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CreateGrantResponse response = kmsClient.createGrant(grantRequest);
        return response.grantId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void enableKeyRotation(KmsClient kmsClient, String keyId) {
    try {
        EnableKeyRotationRequest enableKeyRotationRequest =
        EnableKeyRotationRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.enableKeyRotation(enableKeyRotationRequest);
    }
}
```

```
        System.out.println("Key rotation has been enabled for key with id [" +
keyId + "]);

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,
String aliasName) {
        try {
            CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
                .aliasName(aliasName)
                .targetKeyId(targetKeyId)
                .build();

            kmsClient.createAlias(aliasRequest);
            System.out.println(aliasName + " was successfully created.");

        } catch (ResourceExistsException e) {
            System.err.println("Alias already exists: " + e.getMessage());
            System.err.println("Moving on...");
        } catch (Exception e) {
            System.err.println("An unexpected error occurred: " + e.getMessage());
            System.err.println("Moving on...");
        }
    }

    public static SdkBytes encryptData(KmsClient kmsClient, String keyId, String
text) {
        try {
            SdkBytes myBytes = SdkBytes.fromUtf8String(text);
            EncryptRequest encryptRequest = EncryptRequest.builder()
                .keyId(keyId)
                .plaintext(myBytes)
                .build();

            EncryptResponse response = kmsClient.encrypt(encryptRequest);
            String algorithm = response.encryptionAlgorithm().toString();
            System.out.println("The string was encrypted with algorithm " +
algorithm + ".");

            // Get the encrypted data.
```

```
        SdkBytes encryptedData = response.ciphertextBlob();
        return encryptedData;

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return null;
}

public static String createKey(KmsClient kmsClient, String keyDesc) {
    try {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
            .keyUsage("ENCRYPT_DECRYPT")
            .build();

        CreateKeyResponse result = kmsClient.createKey(keyRequest);
        System.out.println("Symmetric key with ARN [" +
result.keyMetadata().arn() + "] has been created.");
        return result.keyMetadata().keyId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Enable the KMS key.
public static void enableKey(KmsClient kmsClient, String keyId) {
    try {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.enableKey(enableKeyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        System.out.println("");
        System.out.println("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            System.out.println("Continuing with the program...");
            System.out.println("");
            break;
        } else {
            // Handle invalid input.
            System.out.println("Invalid input. Please try again.");
        }
    }
}

private static String getAccountId(){
    try (StsClient stsClient = StsClient.create()){
        GetCallerIdentityResponse callerIdentity =
stsClient.getCallerIdentity();
        return callerIdentity.account();
    }
}
}
```

- APIの詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [CreateKey](#)
  - [DescribeKey](#)
  - [DisableKey](#)
  - [EnableKey](#)
  - [GenerateDataKey](#)
  - [ListKeys](#)
  - [ScheduleKeyDeletion](#)
  - [Sign](#)

## SDK for Java 2.x を使用した Lambda の例

次のコード例は、Lambda AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### 開始方法

#### Hello Lambda

次のコード例では、Lambda の使用を開始する方法について示しています。

#### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
package com.example.lambda;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.ListFunctionsResponse;
import software.amazon.awssdk.services.lambda.model.FunctionConfiguration;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListLambdaFunctions {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        listFunctions(awsLambda);
        awsLambda.close();
    }

    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config : list) {
                System.out.println("The function name is " + config.functionName());
            }
        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [ListFunctions](#)」の「」を参照してください。AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)
- [サーバーレスサンプル](#)

## アクション

### CreateFunction

次の例は、CreateFunction を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.CreateFunctionRequest;
import software.amazon.awssdk.services.lambda.model.FunctionCode;
import software.amazon.awssdk.services.lambda.model.CreateFunctionResponse;
import software.amazon.awssdk.services.lambda.model.GetFunctionRequest;
import software.amazon.awssdk.services.lambda.model.GetFunctionResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.Runtime;
import software.amazon.awssdk.services.lambda.waiters.LambdaWaiter;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;

/**
 * This code example requires a ZIP or JAR that represents the code of the
 * Lambda function.
 * If you do not have a ZIP or JAR, please refer to the following document:
 *
 * https://github.com/aws-doc-sdk-examples/tree/master/javav2/usecases/
creating_workflows_stepfunctions
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```



```
*/

public class CreateFunction {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <functionName> <filePath> <role> <handler>\s

            Where:
                functionName - The name of the Lambda function.\s
                filePath - The path to the ZIP or JAR where the code is located.
\s
                role - The role ARN that has Lambda permissions.\s
                handler - The fully qualified method name (for example,
example.Handler::handleRequest). \s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        String role = args[2];
        String handler = args[3];
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        createLambdaFunction(awsLambda, functionName, filePath, role, handler);
        awsLambda.close();
    }

    public static void createLambdaFunction(LambdaClient awsLambda,
        String functionName,
        String filePath,
        String role,
        String handler) {

        try {
```

```
LambdaWaiter waiter = awsLambda.waiter();
InputStream is = new FileInputStream(filePath);
SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

FunctionCode code = FunctionCode.builder()
    .zipFile(fileToUpload)
    .build();

CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
    .functionName(functionName)
    .description("Created by the Lambda Java API")
    .code(code)
    .handler(handler)
    .runtime(Runtime.JAVA8)
    .role(role)
    .build();

// Create a Lambda function using a waiter.
CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
    .functionName(functionName)
    .build();

WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("The function ARN is " +
functionResponse.functionArn());

    } catch (LambdaException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、「APIリファレンス[CreateFunction](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteFunction

次の例は、DeleteFunction を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.DeleteFunctionRequest;
import software.amazon.awssdk.services.lambda.model.LambdaException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteFunction {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <functionName>\s

                Where:
                functionName - The name of the Lambda function.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        Region region = Region.US_EAST_1;
        LambdaClient awsLambda = LambdaClient.builder()
                .region(region)
                .build();
```

```
        deleteLambdaFunction(awsLambda, functionName);
        awsLambda.close();
    }

    public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                .functionName(functionName)
                .build();

            awsLambda.deleteFunction(request);
            System.out.println("The " + functionName + " function was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [DeleteFunction](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## Invoke

次の例は、Invoke を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
import org.json.JSONObject;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.lambda.LambdaClient;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.InvokeRequest;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lambda.model.InvokeResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;

public class LambdaInvoke {

    /**
     * Function names appear as
     * arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
     * you can retrieve the value by looking at the function in the AWS Console
     *
     * Also, set up your development environment, including your credentials.
     *
     * For information, see this documentation topic:
     *
     * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
     */

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <functionName>\s

            Where:
                functionName - The name of the Lambda function\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        invokeFunction(awsLambda, functionName);
        awsLambda.close();
    }
}
```

```
}

public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res = null;
    try {
        // Need a SdkBytes instance for the payload.
        JSONObject jsonObj = new JSONObject();
        jsonObj.put("inputValue", "2000");
        String json = jsonObj.toString();
        SdkBytes payload = SdkBytes.fromUtf8String(json);

        // Setup an InvokeRequest.
        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
            .build();

        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String();
        System.out.println(value);

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[Invoke](#)」を参照してください。

## シナリオ

### 関数の使用を開始します


次のコードサンプルは、以下の操作方法を示しています。

- IAM ロールと Lambda 関数を作成し、ハンドラーコードをアップロードします。
- 1 つのパラメーターで関数を呼び出して、結果を取得します。
- 関数コードを更新し、環境変数で設定します。

- 新しいパラメーターで関数を呼び出して、結果を取得します。返された実行ログを表示します。
- アカウントの関数を一覧表示し、リソースをクリーンアップします。

詳細については、「[コンソールで Lambda 関数を作成する](#)」を参照してください。

SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/*
 * Lambda function names appear as:
 *
 * arn:aws:lambda:us-west-2:335556666777:function>HelloFunction
 *
 * To find this value, look at the function in the AWS Management Console.
 *
 * Before running this Java code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example performs the following tasks:
 *
 * 1. Creates an AWS Lambda function.
 * 2. Gets a specific AWS Lambda function.
 * 3. Lists all Lambda functions.
 * 4. Invokes a Lambda function.
 * 5. Updates the Lambda function code and invokes it again.
 * 6. Updates a Lambda function's configuration value.
 * 7. Deletes a Lambda function.
 */

public class LambdaScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
```

```
public static void main(String[] args) throws InterruptedException {
    final String usage = ""

        Usage:
            <functionName> <filePath> <role> <handler> <bucketName> <key>\s

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

    if (args.length != 6) {
        System.out.println(usage);
        System.exit(1);
    }

    String functionName = args[0];
    String filePath = args[1];
    String role = args[2];
    String handler = args[3];
    String bucketName = args[4];
    String key = args[5];

    Region region = Region.US_WEST_2;
    LambdaClient awsLambda = LambdaClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the AWS Lambda example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Create an AWS Lambda function.");
```



```
String funArn = createLambdaFunction(awsLambda, functionName, filePath,
role, handler);
System.out.println("The AWS Lambda ARN is " + funArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Get the " + functionName + " AWS Lambda function.");
getFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. List all AWS Lambda functions.");
listFunctions(awsLambda);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Invoke the Lambda function.");
System.out.println("*** Sleep for 1 min to get Lambda function ready.");
Thread.sleep(60000);
invokeFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Update the Lambda function code and invoke it
again.");
updateFunctionCode(awsLambda, functionName, bucketName, key);
System.out.println("*** Sleep for 1 min to get Lambda function ready.");
Thread.sleep(60000);
invokeFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Update a Lambda function's configuration value.");
updateFunctionConfiguration(awsLambda, functionName, handler);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the AWS Lambda function.");
LambdaScenario.deleteLambdaFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The AWS Lambda scenario completed successfully");
System.out.println(DASHES);
```

```
        awsLambda.close();
    }

    public static String createLambdaFunction(LambdaClient awsLambda,
        String functionName,
        String filePath,
        String role,
        String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

            FunctionCode code = FunctionCode.builder()
                .zipFile(fileToUpload)
                .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA8)
                .role(role)
                .build();

            // Create a Lambda function using a waiter
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();

            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            return functionResponse.functionArn();

        } catch (LambdaException | FileNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

```
public static void getFunction(LambdaClient awsLambda, String functionName) {
    try {
        GetFunctionRequest functionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();

        GetFunctionResponse response = awsLambda.getFunction(functionRequest);
        System.out.println("The runtime of this Lambda function is " +
response.configuration().runtime());

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listFunctions(LambdaClient awsLambda) {
    try {
        ListFunctionsResponse functionResult = awsLambda.listFunctions();
        List<FunctionConfiguration> list = functionResult.functions();
        for (FunctionConfiguration config : list) {
            System.out.println("The function name is " + config.functionName());
        }

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res;
    try {
        // Need a SdkBytes instance for the payload.
        JSONObject jsonObj = new JSONObject();
        jsonObj.put("inputValue", "2000");
        String json = jsonObj.toString();
        SdkBytes payload = SdkBytes.fromUtf8String(json);

        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
    }
```

```
        .build();

        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String();
        System.out.println(value);

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateFunctionCode(LambdaClient awsLambda, String
functionName, String bucketName, String key) {
    try {
        LambdaWaiter waiter = awsLambda.waiter();
        UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
            .functionName(functionName)
            .publish(true)
            .s3Bucket(bucketName)
            .s3Key(key)
            .build();

        UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest);
        GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
            .functionName(functionName)
            .build();

        WaiterResponse<GetFunctionConfigurationResponse> waiterResponse = waiter
            .waitUntilFunctionUpdated(getFunctionConfigRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The last modified value is " +
response.lastModified());

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
    try {
        UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
            .functionName(functionName)
            .handler(handler)
            .runtime(Runtime.JAVA11)
            .build();

        awsLambda.updateFunctionConfiguration(configurationRequest);

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("The " + functionName + " function was deleted");

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Invoke](#)

- [ListFunctions](#)
- [UpdateFunctionCode](#)
- [UpdateFunctionConfiguration](#)

## サーバーレスサンプル

Kinesis トリガーから Lambda 関数を呼び出す

次のコード例では、Kinesis ストリームからレコードを受信することによってトリガーされるイベントを受け取る、Lambda 関数の実装方法を示しています。この関数は Kinesis ペイロードを取得し、それを Base64 からデコードして、そのレコードの内容をログ記録します。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [サーバーレスサンプル](#) リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Java を使用した Lambda での Kinesis イベントの消費。

```
// 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;
}
}
```

## Amazon S3 トリガーから Lambda 関数を呼び出す

次のコード例は、S3 バケットにオブジェクトをアップロードすることによってトリガーされるイベントを受け取る Lambda 関数を実装する方法を示しています。この関数は、イベントパラメータから S3 バケット名とオブジェクトキーを取得し、Amazon S3 API を呼び出してオブジェクトのコンテンツタイプを取得してログに記録します。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [サーバーレスサンプル](#) リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Java を使用して Lambda で S3 イベントを消費します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
```

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

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
    com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotificat

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
            String srcBucket = record.getS3().getBucket().getName();
            String srcKey = record.getS3().getObject().getUrlDecodedKey();

            S3Client s3Client = S3Client.builder().build();
            HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucket)
            .key(key)
            .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```



## Amazon SNS トリガーから Lambda 関数を呼び出す

次のコード例は、SNS トピックからメッセージを受信することによってトリガーされるイベントを受け取る Lambda 関数を実装する方法を示しています。この関数はイベントパラメータからメッセージを取得し、各メッセージの内容を記録します。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [サーバーレスサンプル](#) リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Java を使用した Lambda での SNS イベントの消費。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;

import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
    }
}
```

```
    }
    return Boolean.TRUE;
}

public void processRecord(SNSRecord record) {
    try {
        String message = record.getSNS().getMessage();
        logger.log("message: " + message);
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
}
```

## Amazon SQS トリガーから Lambda 関数を呼び出す

次のコード例では、SQS キューからメッセージを受信することによってトリガーされるイベントを受け取る、Lambda 関数の実装方法を示しています。この関数はイベントパラメータからメッセージを取得し、各メッセージの内容を記録します。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [サーバーレスサンプル](#) リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Java を使用した Lambda での SQS イベントの消費。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;
```

```
public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }
    }
}
```

## Kinesis トリガーを使用した Lambda 関数でのバッチアイテムの失敗のレポート

以下のコード例では、Kinesis ストリームからイベントを受け取る Lambda 関数のための、部分的なバッチレスポンスの実装方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [サーバーレスサンプル](#) リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Java を使用した Lambda での Kinesis バッチアイテム失敗のレポート。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
ArrayList<>();
        String curRecordSequenceNumber = "";

        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
immediately.
                Lambda will immediately begin to retry processing from this
failed item onwards. */
                batchItemFailures.add(new
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

        return new StreamsEventResponse(batchItemFailures);
    }
}
```

## DynamoDB トリガーで Lambda 関数のバッチアイテムの失敗をレポートする

次のコード例は、DynamoDB ストリームからイベントを受信する Lambda 関数に部分的なバッチレスポンスを実装する方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [サーバーレスサンプル](#) リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Java を使用して Lambda で DynamoDB のバッチアイテム失敗のレポート。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
    Serializable> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
    {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
        ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
        input.getRecords()) {
            try {
```

```
        //Process your record
        StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
        curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

    } catch (Exception e) {
        /* Since we are working with streams, we can return the failed item
        immediately.
           Lambda will immediately begin to retry processing from this
        failed item onwards. */
        batchItemFailures.add(new
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
        return new StreamsEventResponse(batchItemFailures);
    }
}

return new StreamsEventResponse();
}
}
```

## Amazon SQS トリガーを使用した Lambda 関数でのバッチアイテムの失敗のレポート

以下のコード例では、SQS キューからイベントを受け取る Lambda 関数のための、部分的なバッチレスポンスの実装方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

### SDK for Java 2.x

#### Note

については、「」を参照してください [GitHub](#)。 [サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

## Java を使用した Lambda での SQS バッチアイテム失敗のレポート。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
```

```
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {

        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
ArrayList<SQSBatchResponse.BatchItemFailure>();
        String messageId = "";
        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
                messageId = message.getMessageId();
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
SQSBatchResponse.BatchItemFailure(messageId));
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }
}
```

## MediaConvert SDK for Java 2.x を使用した の例

次のコード例は、AWS SDK for Java 2.x を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています MediaConvert。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

## トピック


- [アクション](#)

## アクション

### CreateJob

次の例は、CreateJob を使用する方法を説明しています。

#### SDK for Java 2.x

 Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
package com.example.mediaconvert;

import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.Output;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroup;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.HlsGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupType;
import software.amazon.awssdk.services.mediaconvert.model.HlsDirectoryStructure;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestDurationFormat;
import software.amazon.awssdk.services.mediaconvert.model.HlsStreamInfResolution;
import software.amazon.awssdk.services.mediaconvert.model.HlsClientCache;
import software.amazon.awssdk.services.mediaconvert.model.HlsCaptionLanguageSetting;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestCompression;
import software.amazon.awssdk.services.mediaconvert.model.HlsCodecSpecification;
import software.amazon.awssdk.services.mediaconvert.model.HlsOutputSelection;
import software.amazon.awssdk.services.mediaconvert.model.HlsProgramDateTime;
```



```
import software.amazon.awssdk.services.mediaconvert.model.HlsTimedMetadataId3Frame;
import software.amazon.awssdk.services.mediaconvert.model.HlsSegmentControl;
import software.amazon.awssdk.services.mediaconvert.model.FileGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.ContainerSettings;
import software.amazon.awssdk.services.mediaconvert.model.VideoDescription;
import software.amazon.awssdk.services.mediaconvert.model.ContainerType;
import software.amazon.awssdk.services.mediaconvert.model.ScalingBehavior;
import software.amazon.awssdk.services.mediaconvert.model.VideoTimecodeInsertion;
import software.amazon.awssdk.services.mediaconvert.model.ColorMetadata;
import software.amazon.awssdk.services.mediaconvert.model.RespondToAfd;
import software.amazon.awssdk.services.mediaconvert.model.AfdSignaling;
import software.amazon.awssdk.services.mediaconvert.model.DropFrameTimecode;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264Settings;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodec;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.H264RateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.H264QualityTuningLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264SceneChangeDetect;
import
    software.amazon.awssdk.services.mediaconvert.model.AacAudioDescriptionBroadcasterMix;
import software.amazon.awssdk.services.mediaconvert.model.H264ParControl;
import software.amazon.awssdk.services.mediaconvert.model.AacRawFormat;
import software.amazon.awssdk.services.mediaconvert.model.H264QvbrSettings;
import
    software.amazon.awssdk.services.mediaconvert.model.H264FramerateConversionAlgorithm;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264FramerateControl;
import software.amazon.awssdk.services.mediaconvert.model.AacCodingMode;
import software.amazon.awssdk.services.mediaconvert.model.H264Telecine;
import
    software.amazon.awssdk.services.mediaconvert.model.H264FlickerAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264GopSizeUnits;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.H264GopBReference;
import software.amazon.awssdk.services.mediaconvert.model.AudioTypeControl;
import software.amazon.awssdk.services.mediaconvert.model.AntiAlias;
import software.amazon.awssdk.services.mediaconvert.model.H264SlowPal;
import
    software.amazon.awssdk.services.mediaconvert.model.H264SpatialAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264Syntax;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Settings;
import software.amazon.awssdk.services.mediaconvert.model.InputDenoiseFilter;
```

```
import
    software.amazon.awssdk.services.mediaconvert.model.H264TemporalAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobResponse;
import
    software.amazon.awssdk.services.mediaconvert.model.H264UnregisteredSeiTimecode;
import software.amazon.awssdk.services.mediaconvert.model.H264EntropyEncoding;
import software.amazon.awssdk.services.mediaconvert.model.InputPsiControl;
import software.amazon.awssdk.services.mediaconvert.model.ColorSpace;
import software.amazon.awssdk.services.mediaconvert.model.H264RepeatPps;
import software.amazon.awssdk.services.mediaconvert.model.H264FieldEncoding;
import software.amazon.awssdk.services.mediaconvert.model.M3u8NielsenId3;
import software.amazon.awssdk.services.mediaconvert.model.InputDeblockFilter;
import software.amazon.awssdk.services.mediaconvert.model.InputRotate;
import software.amazon.awssdk.services.mediaconvert.model.H264DynamicSubGop;
import software.amazon.awssdk.services.mediaconvert.model.TimedMetadata;
import software.amazon.awssdk.services.mediaconvert.model.JobSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioDefaultSelection;
import software.amazon.awssdk.services.mediaconvert.model.VideoSelector;
import software.amazon.awssdk.services.mediaconvert.model.AacSpecification;
import software.amazon.awssdk.services.mediaconvert.model.Input;
import software.amazon.awssdk.services.mediaconvert.model.OutputSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264AdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.AudioLanguageCodeControl;
import software.amazon.awssdk.services.mediaconvert.model.InputFilterEnable;
import software.amazon.awssdk.services.mediaconvert.model.AudioDescription;
import software.amazon.awssdk.services.mediaconvert.model.H264InterlaceMode;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.AacSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodec;
import software.amazon.awssdk.services.mediaconvert.model.AacRateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.AacCodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.HlsIFrameOnlyManifest;
import software.amazon.awssdk.services.mediaconvert.model.FrameCaptureSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioSelector;
import software.amazon.awssdk.services.mediaconvert.model.M3u8PcrControl;
import software.amazon.awssdk.services.mediaconvert.model.InputTimecodeSource;
import software.amazon.awssdk.services.mediaconvert.model.HlsSettings;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Scte35Source;

/**
 * Create a MediaConvert job. Must supply MediaConvert access role Amazon
 * Resource Name (ARN), and a
 * valid video input file via Amazon S3 URL.
 *
 */
```

```
* Also, set up your development environment, including your credentials.
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
*/
public class CreateJob {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <mcRoleARN> <fileInput>\s

            Where:
                mcRoleARN - The MediaConvert Role ARN.\s
                fileInput - The URL of an Amazon S3 bucket
where the input file is located.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String mcRoleARN = args[0];
        String fileInput = args[1];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
            .region(region)
            .build();

        String id = createMediaJob(mc, mcRoleARN, fileInput);
        System.out.println("MediaConvert job created. Job Id = " + id);
        mc.close();
    }

    public static String createMediaJob(MediaConvertClient mc, String mcRoleARN,
String fileInput) {

        String s3path = fileInput.substring(0, fileInput.lastIndexOf('/') +
1) + "javasdk/out/";
        String fileOutput = s3path + "index";
        String thumbsOutput = s3path + "thumbs/";
```

```
String mp4Output = s3path + "mp4/";

try {
    DescribeEndpointsResponse res = mc

.describeEndpoints(DescribeEndpointsRequest.builder().maxResults(20).build());

    if (res.endpoints().size() <= 0) {
        System.out.println("Cannot find MediaConvert service
endpoint URL!");

        System.exit(1);
    }
    String endpointURL = res.endpoints().get(0).url();
    System.out.println("MediaConvert service URL: " +
endpointURL);

    System.out.println("MediaConvert role arn: " + mcRoleARN);
    System.out.println("MediaConvert input file: " + fileInput);
    System.out.println("MediaConvert output path: " + s3path);

    MediaConvertClient emc = MediaConvertClient.builder()
        .region(Region.US_WEST_2)
        .endpointOverride(URI.create(endpointURL))
        .build();

    // output group Preset HLS low profile
    Output hlsLow = createOutput("hls_low", "_low", "_$dt$",
750000, 7, 1920, 1080, 640);
    // output group Preset HLS media profile
    Output hlsMedium = createOutput("hls_medium", "_medium", "_
$dt$", 1200000, 7, 1920, 1080, 1280);
    // output group Preset HLS high profole
    Output hlsHigh = createOutput("hls_high", "_high", "_$dt$",
3500000, 8, 1920, 1080, 1920);

    OutputGroup appleHLS = OutputGroup.builder().name("Apple
HLS").customName("Example")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.HLS_GROUP_SETTINGS)

.hlsGroupSettings(HlsGroupSettings.builder()

.directoryStructure(
```

```
HlsDirectoryStructure.SINGLE_DIRECTORY)

.manifestDurationFormat(

    HlsManifestDurationFormat.INTEGER)

.streamInfResolution(

    HlsStreamInfResolution.INCLUDE)

.clientCache(HlsClientCache.ENABLED)

.captionLanguageSetting(

    HlsCaptionLanguageSetting.OMIT)

.manifestCompression(

    HlsManifestCompression.NONE)

.codecSpecification(

    HlsCodecSpecification.RFC_4281)

.outputSelection(

    HlsOutputSelection.MANIFESTS_AND_SEGMENTS)

.programDateTime(HlsProgramDateTime.EXCLUDE)

.programDateTimePeriod(600)

.timedMetadataId3Frame(

    HlsTimedMetadataId3Frame.PRIV)

.timedMetadataId3Period(10)

.destination(fileOutput)

.segmentControl(HlsSegmentControl.SEGMENTED_FILES)

.minFinalSegmentLength((double) 0)
```

```
.segmentLength(4).minSegmentLength(0).build()
                                .build()
                                .outputs(hlsLow, hlsMedium,
hlsHigh).build();

        OutputGroup fileMp4 = OutputGroup.builder().name("File
Group").customName("mp4")

        .outputGroupSettings(OutputGroupSettings.builder()

        .type(OutputGroupType.FILE_GROUP_SETTINGS)

        .fileGroupSettings(FileGroupSettings.builder()

        .destination(mp4Output).build()
                                .build()
                                .outputs(Output.builder().extension("mp4")

        .containerSettings(ContainerSettings.builder()

        .container(ContainerType.MP4).build()

        .videoDescription(VideoDescription.builder().width(1280)
                                .height(720)

        .scalingBehavior(ScalingBehavior.DEFAULT)

        .sharpness(50).antiAlias(AntiAlias.ENABLED)

        .timecodeInsertion(
            VideoTimecodeInsertion.DISABLED)

        .colorMetadata(ColorMetadata.INSERT)

        .respondToAfd(RespondToAfd.NONE)

        .afdSignaling(AfdSignaling.NONE)

        .dropFrameTimecode(DropFrameTimecode.ENABLED)

        .codecSettings(VideoCodecSettings.builder()
```

```
.codec(VideoCodec.H_264)

.h264Settings(H264Settings

    .builder()

    .rateControlMode(

        H264RateControlMode.QVBR)

    .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)

    .qualityTuningLevel(

        H264QualityTuningLevel.SINGLE_PASS)

    .qvbrSettings(

        H264QvbrSettings.builder()

            .qvbrQualityLevel(

                8)

            .build())

    .codecLevel(H264CodecLevel.AUTO)

    .codecProfile(H264CodecProfile.MAIN)

    .maxBitrate(2400000)

    .framerateControl(

        H264FramerateControl.INITIALIZE_FROM_SOURCE)

    .gopSize(2.0)

    .gopSizeUnits(H264GopSizeUnits.SECONDS)

    .numberBframesBetweenReferenceFrames(

        2)
```

```
.gopClosedCadence(  
    1)  
.gopBReference(H264GopBReference.DISABLED)  
.slowPal(H264SlowPal.DISABLED)  
.syntax(H264Syntax.DEFAULT)  
.numberReferenceFrames(  
    3)  
.dynamicSubGop(H264DynamicSubGop.STATIC)  
.fieldEncoding(H264FieldEncoding.PAFF)  
.sceneChangeDetect(  
    H264SceneChangeDetect.ENABLED)  
.minIInterval(0)  
.telecine(H264Telecine.NONE)  
.framerateConversionAlgorithm(  
    H264FramerateConversionAlgorithm.DUPLICATE_DROP)  
.entropyEncoding(  
    H264EntropyEncoding.CABAC)  
.slices(1)  
.unregisteredSeiTimecode(  
    H264UnregisteredSeiTimecode.DISABLED)  
.repeatPps(H264RepeatPps.DISABLED)  
.adaptiveQuantization(  

```



```
                H264AdaptiveQuantization.HIGH)

                .spatialAdaptiveQuantization(

                    H264SpatialAdaptiveQuantization.ENABLED)

                .temporalAdaptiveQuantization(

                    H264TemporalAdaptiveQuantization.ENABLED)

                .flickerAdaptiveQuantization(

                    H264FlickerAdaptiveQuantization.DISABLED)

                .softness(0)

                .interlaceMode(H264InterlaceMode.PROGRESSIVE)

                .build())

        .build()

        .build()

        .audioDescriptions(AudioDescription.builder())

        .audioTypeControl(AudioTypeControl.FOLLOW_INPUT)

        .languageCodeControl(

            AudioLanguageCodeControl.FOLLOW_INPUT)

        .codecSettings(AudioCodecSettings.builder())

            .codec(AudioCodec.AAC)

            .aacSettings(AacSettings

                .builder()

                    .codecProfile(AacCodecProfile.LC)

                    .rateControlMode(
```

```
                AacRateControlMode.CBR)

                .codingMode(AacCodingMode.CODING_MODE_2_0)

                .sampleRate(44100)

                .bitrate(160000)

                .rawFormat(AacRawFormat.NONE)

                .specification(AacSpecification.MPEG4)

                .audioDescriptionBroadcasterMix(

                    AacAudioDescriptionBroadcasterMix.NORMAL)

                .build()

            .build()

                .build()

                .build()

                .build();
        OutputGroup thumbs = OutputGroup.builder().name("File
Group").customName("thumbs")

        .outputGroupSettings(OutputGroupSettings.builder()

        .type(OutputGroupType.FILE_GROUP_SETTINGS)

        .fileGroupSettings(FileGroupSettings.builder()

        .destination(thumbsOutput).build()

                .build()

                .outputs(Output.builder().extension("jpg")

        .containerSettings(ContainerSettings.builder()

        .container(ContainerType.RAW).build()

        .videoDescription(VideoDescription.builder()

        .scalingBehavior(ScalingBehavior.DEFAULT)
```

```

    .sharpness(50).antiAlias(AntiAlias.ENABLED)

    .timecodeInsertion(
        VideoTimecodeInsertion.DISABLED)

    .colorMetadata(ColorMetadata.INSERT)

    .dropFrameTimecode(DropFrameTimecode.ENABLED)

    .codecSettings(VideoCodecSettings.builder()

        .codec(VideoCodec.FRAME_CAPTURE)

        .frameCaptureSettings(
            FrameCaptureSettings
                .builder()

                .framerateNumerator(
                    1)

                .framerateDenominator(
                    1)

                .maxCaptures(10000000)

                .quality(80)

                .build())

        .build())

        .build()

        .build()

        .build();

    Map<String, AudioSelector> audioSelectors = new HashMap<>();
    audioSelectors.put("Audio Selector 1",
        AudioSelector.builder().defaultSelection(AudioDefaultSelection.DEFAULT)

```

```

        .offset(0).build());

        JobSettings jobSettings =
JobSettings.builder().inputs(Input.builder()
                                .audioSelectors(audioSelectors)
                                .videoSelector(
VideoSelector.builder().colorSpace(ColorSpace.FOLLOW)
                .rotate(InputRotate.DEGREE_0).build())
                .filterEnable(InputFilterEnable.AUTO).filterStrength(0)
                    .deblockFilter(InputDeblockFilter.DISABLED)
                .denoiseFilter(InputDenoiseFilter.DISABLED).psiControl(InputPsiControl.USE_PSI)
                .timecodeSource(InputTimecodeSource.EMBEDDED).fileInput(fileInput).build())
                    .outputGroups(appleHLS, thumbs,
fileMp4).build());

        CreateJobRequest createJobRequest =
CreateJobRequest.builder().role(mcRoleARN)
                    .settings(jobSettings)
                    .build();

        CreateJobResponse createJobResponse =
emc.createJob(createJobRequest);
        return createJobResponse.job().id();

    } catch (MediaConvertException e) {
        System.out.println(e.toString());
        System.exit(0);
    }
    return "";
}

private final static Output createOutput(String customName,
        String nameModifier,
        String segmentModifier,
        int qvbrMaxBitrate,
        int qvbrQualityLevel,
        int originWidth,
        int originHeight,
        int targetWidth) {

```

```

        int targetHeight = Math.round(originHeight * targetWidth /
originWidth)
                                - (Math.round(originHeight * targetWidth /
originWidth) % 4);
        Output output = null;
        try {
            output =
Output.builder().nameModifier(nameModifier).outputSettings(OutputSettings.builder()

.hlsSettings(HlsSettings.builder().segmentModifier(segmentModifier)

.audioGroupId("program_audio")

.iFrameOnlyManifest(HlsIFrameOnlyManifest.EXCLUDE).build())
                                .build())

.containerSettings(ContainerSettings.builder().container(ContainerType.M3_U8)

.m3u8Settings(M3u8Settings.builder().audioFramesPerPes(4)

.pcrControl(M3u8PcrControl.PCR_EVERY_PES_PACKET)

.pmtPid(480).privateMetadataPid(503)

.programNumber(1).patInterval(0).pmtInterval(0)

.scte35Source(M3u8Scte35Source.NONE)

.scte35Pid(500).nielsenId3(M3u8NielsenId3.NONE)

.timedMetadata(TimedMetadata.NONE)

.timedMetadataPid(502).videoPid(481)

.audioPids(482, 483, 484, 485, 486, 487, 488,

            489, 490, 491, 492)

                                .build())

                                .build())
                                .videoDescription(
VideoDescription.builder().width(targetWidth)

```

```
.height(targetHeight)

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(
    VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()
    .codec(VideoCodec.H_264)
    .h264Settings(H264Settings
        .builder()
        .rateControlMode(
            H264RateControlMode.QVBR)
        .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
        .qualityTuningLevel(
            H264QualityTuningLevel.SINGLE_PASS)
        .qvbrSettings(H264QvbrSettings
            .builder()
            .qvbrQualityLevel(
                qvbrQualityLevel)
```

```
        .build()

        .codecLevel(H264CodecLevel.AUTO)

        .codecProfile((targetHeight > 720

            && targetWidth > 1280)

            ? H264CodecProfile.HIGH

            : H264CodecProfile.MAIN)

        .maxBitrate(qvbrMaxBitrate)

        .framerateControl(

            H264FramerateControl.INITIALIZE_FROM_SOURCE)

        .gopSize(2.0)

        .gopSizeUnits(H264GopSizeUnits.SECONDS)

        .numberBFramesBetweenReferenceFrames(

            2)

        .gopClosedCadence(

            1)

        .gopBReference(H264GopBReference.DISABLED)

        .slowPal(H264SlowPal.DISABLED)

        .syntax(H264Syntax.DEFAULT)

        .numberReferenceFrames(

            3)

        .dynamicSubGop(H264DynamicSubGop.STATIC)

        .fieldEncoding(H264FieldEncoding.PAFF)
```

```
.sceneChangeDetect(  
    H264SceneChangeDetect.ENABLED)  
  
.minIInterval(0)  
  
.telecine(H264Telecine.NONE)  
  
.framerateConversionAlgorithm(  
    H264FramerateConversionAlgorithm.DUPLICATE_DROP)  
  
.entropyEncoding(  
    H264EntropyEncoding.CABAC)  
  
.slices(1)  
  
.unregisteredSeiTimecode(  
    H264UnregisteredSeiTimecode.DISABLED)  
  
.repeatPps(H264RepeatPps.DISABLED)  
  
.adaptiveQuantization(  
    H264AdaptiveQuantization.HIGH)  
  
.spatialAdaptiveQuantization(  
    H264SpatialAdaptiveQuantization.ENABLED)  
  
.temporalAdaptiveQuantization(  
    H264TemporalAdaptiveQuantization.ENABLED)  
  
.flickerAdaptiveQuantization(  
    H264FlickerAdaptiveQuantization.DISABLED)  
  
.softness(0)  
  
.interlaceMode(H264InterlaceMode.PROGRESSIVE)
```



```

        .build()

        .build()

        .build()

        .audioDescriptions(AudioDescription.builder()

        .audioTypeControl(AudioTypeControl.FOLLOW_INPUT)

        .languageCodeControl(AudioLanguageCodeControl.FOLLOW_INPUT)

        .codecSettings(AudioCodecSettings.builder()

        .codec(AudioCodec.AAC).aacSettings(AacSettings

        .builder()

        .codecProfile(AacCodecProfile.LC)

        .rateControlMode(

                AacRateControlMode.CBR)

        .codingMode(AacCodingMode.CODING_MODE_2_0)

        .sampleRate(44100)

        .bitrate(96000)

        .rawFormat(AacRawFormat.NONE)

        .specification(AacSpecification.MPEG4)

        .audioDescriptionBroadcasterMix(

                AacAudioDescriptionBroadcasterMix.NORMAL)

        .build()

        .build()

        .build()

        .build();
    } catch (MediaConvertException e) {
        e.printStackTrace();
    }

```

```
        System.exit(0);
    }
    return output;
}
}
```

- API の詳細については、「API リファレンス [CreateJob](#)」の「」を参照してください。AWS SDK for Java 2.x

## GetJob

次の例は、GetJob を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.GetJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.GetJobResponse;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import java.net.URI;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetJob {
```

```
public static void main(String[] args) {

    final String usage = "\n" +
        " <jobId> \n\n" +
        "Where:\n" +
        "  jobId - The job id value.\n\n";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String jobId = args[0];
    Region region = Region.US_WEST_2;
    MediaConvertClient mc = MediaConvertClient.builder()
        .region(region)
        .build();

    getSpecificJob(mc, jobId);
    mc.close();
}

public static void getSpecificJob(MediaConvertClient mc, String jobId) {
    try {
        DescribeEndpointsResponse res =
mc.describeEndpoints(DescribeEndpointsRequest.builder()
            .maxResults(20)
            .build());

        if (res.endpoints().size() <= 0) {
            System.out.println("Cannot find MediaConvert service endpoint
URL!");
            System.exit(1);
        }
        String endpointURL = res.endpoints().get(0).url();
        MediaConvertClient emc = MediaConvertClient.builder()
            .region(Region.US_WEST_2)
            .endpointOverride(URI.create(endpointURL))
            .build();

        GetJobRequest jobRequest = GetJobRequest.builder()
            .id(jobId)
            .build();
```

```
        GetJobResponse response = emc.getJob(jobRequest);
        System.out.println("The ARN of the job is " + response.job().arn());

    } catch (MediaConvertException e) {
        System.out.println(e.toString());
        System.exit(0);
    }
}
}
```

- API の詳細については、「API リファレンス [GetJob](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListJobs

次の例は、ListJobs を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsResponse;
import software.amazon.awssdk.services.mediaconvert.model.Job;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import java.net.URI;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListJobs {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
            .region(region)
            .build();

        listCompleteJobs(mc);
        mc.close();
    }

    public static void listCompleteJobs(MediaConvertClient mc) {
        try {
            DescribeEndpointsResponse res =
mc.describeEndpoints(DescribeEndpointsRequest.builder()
                .maxResults(20)
                .build());

            if (res.endpoints().size() <= 0) {
                System.out.println("Cannot find MediaConvert service endpoint
URL!");
                System.exit(1);
            }

            String endpointURL = res.endpoints().get(0).url();
            MediaConvertClient emc = MediaConvertClient.builder()
                .region(Region.US_WEST_2)
                .endpointOverride(URI.create(endpointURL))
                .build();

            ListJobsRequest jobsRequest = ListJobsRequest.builder()
                .maxResults(10)
                .status("COMPLETE")
                .build();

            ListJobsResponse jobsResponse = emc.listJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("The JOB ARN is : " + job.arn());
            }
        }
    }
}
```

```
        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
}
```

- API の詳細については、「API リファレンス [ListJobs](#)」の「」を参照してください。AWS SDK for Java 2.x

## SDK for Java 2.x を使用した Migration Hub の例

次のコード例は、Migration Hub AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- [アクション](#)

アクション

### DeleteProgressUpdateStream

次の例は、DeleteProgressUpdateStream を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「API リファレンス [DeleteProgressUpdateStream](#)」の「」を参照してください。AWS SDK for Java 2.x

## DescribeApplicationState

次の例は、DescribeApplicationState を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。



```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [DescribeApplicationState](#)」の「」を参照してください。AWS SDK for Java 2.x

## DescribeMigrationTask

次の例は、DescribeMigrationTask を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
}
}
```

- API の詳細については、「API リファレンス [DescribeMigrationTask](#)」の「」を参照してください。AWS SDK for Java 2.x

## ImportMigrationTask

次の例は、ImportMigrationTask を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [ImportMigrationTask](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListApplications

次の例は、ListApplications を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

```
}  
}
```

- API の詳細については、「API リファレンス [ListApplications](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListCreatedArtifacts

次の例は、ListCreatedArtifacts を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;  
import software.amazon.awssdk.services.migrationhub.model.CreatedArtifact;  
import  
    software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsRequest;  
import  
    software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsResponse;  
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;  
import java.util.List;  
  
/**  
 * To run this Java V2 code example, ensure that you have setup your development  
 * environment, including your credentials.  
 *  
 * For information, see this documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class ListCreatedArtifacts {  
    public static void main(String[] args) {  
        Region region = Region.US_WEST_2;  
        MigrationHubClient migrationClient = MigrationHubClient.builder()
```



```
        .region(region)
        .build();

    listArtifacts(migrationClient);
    migrationClient.close();
}

public static void listArtifacts(MigrationHubClient migrationClient) {
    try {
        ListCreatedArtifactsRequest listCreatedArtifactsRequest =
ListCreatedArtifactsRequest.builder()
            .maxResults(10)
            .migrationTaskName("SampleApp5")
            .progressUpdateStream("ProgressStreamB")
            .build();

        ListCreatedArtifactsResponse response =
migrationClient.listCreatedArtifacts(listCreatedArtifactsRequest);
        List<CreatedArtifact> apps = response.createdArtifactList();
        for (CreatedArtifact artifact : apps) {
            System.out.println("App Id is " + artifact.description());
            System.out.println("The name is " + artifact.name());
        }


    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [ListCreatedArtifacts](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListMigrationTasks

次の例は、ListMigrationTasks を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [ListMigrationTasks](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## SDK for Java 2.x を使用した Amazon Personalize の例

次のコード例は、Amazon Personalize AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### トピック

- [アクション](#)

## アクション

### CreateBatchInferenceJob

次の例は、CreateBatchInferenceJob を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
}
```

- API の詳細については、「API リファレンス [CreateBatchInferenceJob](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateCampaign

次の例は、CreateCampaign を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void createPersonalCampaign(PersonalizeClient personalizeClient,
String solutionVersionArn,
    String name) {

    try {
        CreateCampaignRequest createCampaignRequest =
CreateCampaignRequest.builder()
            .minProvisionedTPS(1)
            .solutionVersionArn(solutionVersionArn)
            .name(name)
            .build();

        CreateCampaignResponse campaignResponse =
personalizeClient.createCampaign(createCampaignRequest);
        System.out.println("The campaign ARN is " +
campaignResponse.campaignArn());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [CreateCampaign](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateDataset

次の例は、CreateDataset を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
}
```


- API の詳細については、「API リファレンス [CreateDataset](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateDatasetExportJob

次の例は、CreateDatasetExportJob を使用する方法を説明しています。



## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
}
```

- API の詳細については、「API リファレンス [CreateDatasetExportJob](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateDatasetGroup

次の例は、CreateDatasetGroup を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
}
```

ドメインデータセットグループを作成します。

```
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 "";  
}
```

- API の詳細については、「API リファレンス [CreateDatasetGroup](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateDatasetImportJob

次の例は、CreateDatasetImportJob を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
}
```

- APIの詳細については、「APIリファレンス[CreateDatasetImportJob](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateEventTracker

次の例は、CreateEventTracker を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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;
}
```

- API の詳細については、「API リファレンス [CreateEventTracker](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateFilter

次の例は、CreateFilter を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
}
```

- APIの詳細については、「APIリファレンス [CreateFilter](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateRecommender

次の例は、CreateRecommender を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
}
```

- API の詳細については、「API リファレンス [CreateRecommender](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateSchema

次の例は、CreateSchema を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
}
```

ドメインを使用してスキーマを作成します。

```
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 "";
}
```

```
}
```

- API の詳細については、「API リファレンス [CreateSchema](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateSolution

次の例は、CreateSolution を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
}
```

- API の詳細については、「API リファレンス [CreateSolution](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateSolutionVersion

次の例は、CreateSolutionVersion を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
}
```

- API の詳細については、「API リファレンス [CreateSolutionVersion](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteCampaign

次の例は、DeleteCampaign を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {

    try {
        DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        personalizeClient.deleteCampaign(campaignRequest);

    } catch (PersonalizeException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DeleteCampaign](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteEventTracker

次の例は、DeleteEventTracker を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void deleteEventTracker(PersonalizeClient personalizeClient,
String eventTrackerArn) {
    try {
        DeleteEventTrackerRequest deleteEventTrackerRequest =
DeleteEventTrackerRequest.builder()
            .eventTrackerArn(eventTrackerArn)
            .build();

        int status =
personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().statusCode

        System.out.println("Status code:" + status);

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



- API の詳細については、「API リファレンス [DeleteEventTracker](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteSolution

次の例は、DeleteSolution を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [DeleteSolution](#)」の「」を参照してください。AWS SDK for Java 2.x

## DescribeCampaign

次の例は、DescribeCampaign を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [DescribeCampaign](#)」の「」を参照してください。 AWS SDK for Java 2.x

## DescribeRecipe

次の例は、DescribeRecipe を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [DescribeRecipe](#)」の「」を参照してください。  
AWS SDK for Java 2.x

**DescribeSolution**

次の例は、DescribeSolution を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [DescribeSolution](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListCampaigns

次の例は、ListCampaigns を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [ListCampaigns](#)」の「」を参照してください。  
AWS SDK for Java 2.x

**ListDatasetGroups**

次の例は、ListDatasetGroups を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [ListDatasetGroups](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListRecipes

次の例は、ListRecipes を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);  
    }  
}
```

- API の詳細については、「API リファレンス [ListRecipes](#)」の「」を参照してください。 AWS SDK for Java 2.x

## ListSolutions

次の例は、ListSolutions を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

- API の詳細については、「API リファレンス [ListSolutions](#)」の「」を参照してください。 AWS SDK for Java 2.x

## UpdateCampaign

次の例は、UpdateCampaign を使用する方法を説明しています。



## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";  
    }
```

- API の詳細については、「API リファレンス [UpdateCampaign](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## SDK for Java 2.x を使用した Amazon Personalize Events の例

次のコード例は、Amazon Personalize Events AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- [アクション](#)

### アクション

#### PutEvents

次の例は、PutEvents を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static int putItems(PersonalizeEventsClient personalizeEventsClient,
    String datasetArn,
    String item1Id,
    String item1PropertyName,
    String item1PropertyValue,
    String item2Id,
    String item2PropertyName,
    String item2PropertyValue) {

    int responseCode = 0;
    ArrayList<Item> items = new ArrayList<>();

    try {
        Item item1 = Item.builder()
            .itemId(item1Id)
            .properties(String.format("{ \"%1$s\": \"%2$s
\}"),
                item1PropertyName,
                item1PropertyValue))
            .build();

        items.add(item1);

        Item item2 = Item.builder()
            .itemId(item2Id)
            .properties(String.format("{ \"%1$s\": \"%2$s
\}"),
                item2PropertyName,
                item2PropertyValue))
            .build();

        items.add(item2);

        PutItemsRequest putItemsRequest = PutItemsRequest.builder()
            .datasetArn(datasetArn)
            .items(items)
            .build();

        responseCode =
personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;
    }
```

```
    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return responseCode;
}
```

- API の詳細については、「API リファレンス [PutEvents](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutUsers

次の例は、PutUsers を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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;
}
```

- API の詳細については、「API リファレンス[PutUsers](#)」の「」を参照してください。AWS SDK for Java 2.x

## SDK for Java 2.x による Amazon Personalize Runtime の例

次のコード例は、Amazon Personalize ランタイム AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- [アクション](#)

## アクション

### GetPersonalizedRanking

次の例は、GetPersonalizedRanking を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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;
}
```

- APIの詳細については、「APIリファレンス[GetPersonalizedRanking](#)」の「」を参照してください。AWS SDK for Java 2.x

## GetRecommendations

次の例は、GetRecommendations を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

推奨アイテムのリストを取得します。

```
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);
}
}
```

ドメインデータセットグループで作成されたレコメンダーから、推奨アイテムのリストを取得します。

```
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);
}
}
```

レコメンデーションをリクエストするときはフィルターを使用してください。

```
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);
}
}
```

- API の詳細については、「API リファレンス [GetRecommendations](#)」の「」を参照してください。AWS SDK for Java 2.x

## SDK for Java 2.x を使用した Amazon Pinpoint の例

次のコード例は、Amazon Pinpoint AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- [アクション](#)

アクション

### CreateApp

次の例は、CreateApp を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
}
}
```

- API の詳細については、「API リファレンス [CreateApp](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateCampaign

次の例は、CreateCampaign を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## キャンペーンの作成

```
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;
}
}
```

- API の詳細については、「API リファレンス [CreateCampaign](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateExportJob

次の例は、CreateExportJob を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## エンドポイントをエクスポートする

```
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);
}
}
}
```

- API の詳細については、「API リファレンス [CreateExportJob](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateImportJob

次の例は、CreateImportJob を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## セグメントのインポート

```
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;
    }
}
```

- APIの詳細については、「API リファレンス [CreateImportJob](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateSegment

次の例は、CreateSegment を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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;
}
}
```

- API の詳細については、「API リファレンス [CreateSegment](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteApp

次の例は、DeleteApp を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

アプリケーションを削除します。

```
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);
    }
}
}
```

- APIの詳細については、「API リファレンス [DeleteApp](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteEndpoint

次の例は、DeleteEndpoint を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

エンドポイントを削除します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteEndpoint {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appName> <endpointId >

            Where:
                appId - The id of the application to delete.
                endpointId - The id of the endpoint to delete.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpointId = args[1];
        System.out.println("Deleting an endpoint with id: " + endpointId);
    }
}
```

```
PinpointClient pinpoint = PinpointClient.builder()
    .region(Region.US_EAST_1)
    .build();

deletePinEndpoint(pinpoint, appId, endpointId);
pinpoint.close();
}

public static void deletePinEndpoint(PinpointClient pinpoint, String appId,
String endpointId) {
    try {
        DeleteEndpointRequest appRequest = DeleteEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpointId)
            .build();

        DeleteEndpointResponse result = pinpoint.deleteEndpoint(appRequest);
        String id = result.endpointResponse().id();
        System.out.println("The deleted endpoint id " + id);

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

- API の詳細については、「API リファレンス[DeleteEndpoint](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## GetEndpoint

次の例は、GetEndpoint を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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");
}
}
```

- APIの詳細については、「APIリファレンス[GetEndpoint](#)」の「」を参照してください。AWS SDK for Java 2.x



## GetSegments

次の例は、GetSegments を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

セグメントを一覧表示します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListSegments {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appId>

            Where:
                appId - The ID of the application that contains a segment.

            """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String appId = args[0];
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    listSegs(pinpoint, appId);
    pinpoint.close();
}

public static void listSegs(PinpointClient pinpoint, String appId) {
    try {
        GetSegmentsRequest request = GetSegmentsRequest.builder()
            .applicationId(appId)
            .build();

        GetSegmentsResponse response = pinpoint.getSegments(request);
        List<SegmentResponse> segments = response.segmentsResponse().item();
        for (SegmentResponse segment : segments) {
            System.out
                .println("Segment " + segment.id() + " " + segment.name() +
                    " " + segment.lastModifiedDate());
        }

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [GetSegments](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## GetSmsChannel

次の例は、GetSmsChannel を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [GetSmsChannel](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## GetUserEndpoints

次の例は、GetUserEndpoints を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [GetUserEndpoints](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## SendMessage

次の例は、SendMessage を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

E メールメッセージを送信します。

```
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.SendMessageRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
```

```
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;

import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendEmailMessage {

    // The character encoding the you want to use for the subject line and
    // message body of the email.
    public static String charset = "UTF-8";

    // The body of the email for recipients whose email clients support HTML
    content.
    static final String body = ""
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS SDK
    for Java 2.x

    """;

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <subject> <appId> <senderAddress>
<toAddress>

            Where:
                subject - The email subject to use.
                senderAddress - The from address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
```



```
        toAddress - The to address. This address has to be verified in Amazon
        Pinpoint in the region you're using to send email\s
        """";

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String subject = args[0];
    String senderAddress = args[1];
    String toAddress = args[2];
    System.out.println("Sending a message");
    PinpointEmailClient pinpoint = PinpointEmailClient.builder()
        .region(Region.US_EAST_1)
        .build();

    sendEmail(pinpoint, subject, senderAddress, toAddress);
    System.out.println("Email was sent");
    pinpoint.close();
}

public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress) {
    try {
        Content content = Content.builder()
            .data(body)
            .build();

        Body messageBody = Body.builder()
            .text(content)
            .build();

        Message message = Message.builder()
            .body(messageBody)
            .subject(Content.builder().data(subject).build())
            .build();

        Destination destination = Destination.builder()
            .toAddresses(toAddress)
            .build();

        EmailContent emailContent = EmailContent.builder()
            .simple(message)
```

```
        .build();

        SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(senderAddress)
            .destination(destination)
            .content(emailContent)
            .build();

        pinpointEmailClient.sendEmail(sendEmailRequest);
        System.out.println("Message Sent");

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

CC の値を使用して E メールメッセージを送信します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendEmailMessageCC {

    // The body of the email.
```

```
static final String body = """"
    Amazon Pinpoint test (AWS SDK for Java 2.x)

    This email was sent through the Amazon Pinpoint Email API using the AWS SDK
for Java 2.x

    """";
public static void main(String[] args) {
    final String usage = """"

        Usage:    <subject> <senderAddress> <toAddress> <ccAddress>

        Where:
            subject - The email subject to use.
            senderAddress - The from address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
            toAddress - The to address. This address has to be verified in Amazon
Pinpoint in the region you're using to send email\s
            ccAddress - The CC address.
        """";

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String subject = args[0];
    String senderAddress = args[1];
    String toAddress = args[2];
    String ccAddress = args[3];

    System.out.println("Sending a message");
    PinpointEmailClient pinpoint = PinpointEmailClient.builder()
        .region(Region.US_EAST_1)
        .build();

    ArrayList<String> ccList = new ArrayList<>();
    ccList.add(ccAddress);
    sendEmail(pinpoint, subject, senderAddress, toAddress, ccList);
    pinpoint.close();
}

public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress, ArrayList<String> ccAddresses) {
```

```
try {
    Content content = Content.builder()
        .data(body)
        .build();

    Body messageBody = Body.builder()
        .text(content)
        .build();

    Message message = Message.builder()
        .body(messageBody)
        .subject(Content.builder().data(subject).build())
        .build();

    Destination destination = Destination.builder()
        .toAddresses(toAddress)
        .ccAddresses(ccAddresses)
        .build();

    EmailContent emailContent = EmailContent.builder()
        .simple(message)
        .build();

    SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
        .fromEmailAddress(senderAddress)
        .destination(destination)
        .content(emailContent)
        .build();

    pinpointEmailClient.sendEmail(sendEmailRequest);
    System.out.println("Message Sent");

} catch (PinpointException e) {
    // Handle exception
    e.printStackTrace();
}
}
```

SMS メッセージを送信します。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessage {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html
    public static String senderId = "MySenderId";

    public static void main(String[] args) {
        final String usage = ""

            Usage:  <message> <appId> <originationNumber>
<destinationNumber>\s

            Where:
```

```
        message - The body of the message to send.
        appId - The Amazon Pinpoint project/application ID
to use when you send this message.
        originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
        destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s
        """;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String message = args[0];
    String appId = args[1];
    String originationNumber = args[2];
    String destinationNumber = args[3];
    System.out.println("Sending a message");
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber);
    pinpoint.close();
}

public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
        String originationNumber,
        String destinationNumber) {
    try {
        Map<String, AddressConfiguration> addressMap = new
HashMap<String, AddressConfiguration>();
        AddressConfiguration addConfig =
AddressConfiguration.builder()
            .channelType(ChannelType.SMS)
            .build();

        addressMap.put(destinationNumber, addConfig);
        SMSMessage smsMessage = SMSMessage.builder()
```

```
        .body(message)
        .messageType(messageType)
        .originationNumber(originationNumber)
        .senderId(senderId)
        .keyword(registeredKeyword)
        .build();

        // Create a DirectMessageConfiguration object.
        DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
        .smsMessage(smsMessage)
        .build();

        MessageRequest msgReq = MessageRequest.builder()
        .addresses(addressMap)
        .messageConfiguration(direct)
        .build();

        // create a SendMessagesRequest object
        SendMessagesRequest request = SendMessagesRequest.builder()
        .applicationId(appId)
        .messageRequest(msgReq)
        .build();

        SendMessagesResponse response =
pinpoint.sendMessage(request);
        MessageResponse msg1 = response.getMessageResponse();
        Map map1 = msg1.getResult();

        // Write out the result of sendMessage.
        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

    } catch (PinpointException e) {
        System.err.println(e.getAwsErrorDetails().getErrorMessage());
        System.exit(1);
    }
}
}
```

SMS メッセージをバッチ送信します。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageBatch {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html
    public static String senderId = "MySenderId";

    public static void main(String[] args) {
        final String usage = ""

            Usage:  <message> <appId> <originationNumber>
<destinationNumber> <destinationNumber1>\s

            Where:
```



```

        message - The body of the message to send.
        appId - The Amazon Pinpoint project/application ID
to use when you send this message.
        originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
        destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).
        destinationNumber1 - The second recipient's phone
number. For best results, you should specify the phone number in E.164 format (for
example, +1-555-555-5654).\s
        """;

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String message = args[0];
    String appId = args[1];
    String originationNumber = args[2];
    String destinationNumber = args[3];
    String destinationNumber1 = args[4];
    System.out.println("Sending a message");
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber, destinationNumber1);
    pinpoint.close();
}

public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
        String originationNumber,
        String destinationNumber, String destinationNumber1) {
    try {
        Map<String, AddressConfiguration> addressMap = new
HashMap<String, AddressConfiguration>();
        AddressConfiguration addConfig =
AddressConfiguration.builder()
            .channelType(ChannelType.SMS)

```

```
                .build());

        // Add an entry to the Map object for each number to whom
you want to send a
        // message.
        addressMap.put(destinationNumber, addConfig);
        addressMap.put(destinationNumber1, addConfig);
        SMSMessage smsMessage = SMSMessage.builder()
                .body(message)
                .messageType(messageType)
                .originationNumber(originationNumber)
                .senderId(senderId)
                .keyword(registeredKeyword)
                .build();

        // Create a DirectMessageConfiguration object.
        DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
                .smsMessage(smsMessage)
                .build();

        MessageRequest msgReq = MessageRequest.builder()
                .addresses(addressMap)
                .messageConfiguration(direct)
                .build();

        // Create a SendMessagesRequest object.
        SendMessagesRequest request = SendMessagesRequest.builder()
                .applicationId(appId)
                .messageRequest(msgReq)
                .build();

        SendMessagesResponse response =
pinpoint.sendMessage(request);
        MessageResponse msg1 = response.messageResponse();
        Map map1 = msg1.result();

        // Write out the result of sendMessage.
        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
}
```

- API の詳細については、「API リファレンス [SendMessages](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## UpdateEndpoint

次の例は、UpdateEndpoint を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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;
}
}
```

- API の詳細については、「API リファレンス [UpdateEndpoint](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## SDK for Java 2.x を使用した Amazon Pinpoint SMS および音声 API の例

次のコード例は、Amazon Pinpoint SMS および音声 API AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- [アクション](#)

### アクション

#### SendVoiceMessage

次の例は、SendVoiceMessage を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpointsmsvoice.PinpointSmsVoiceClient;
import software.amazon.awssdk.services.pinpointsmsvoice.model.SSMLMessageType;
import software.amazon.awssdk.services.pinpointsmsvoice.model.VoiceMessageContent;
import
    software.amazon.awssdk.services.pinpointsmsvoice.model.SendVoiceMessageRequest;
import
    software.amazon.awssdk.services.pinpointsmsvoice.model.PinpointSmsVoiceException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendVoiceMessage {

    // The Amazon Polly voice that you want to use to send the message. For a
    list
    // of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
    static final String voiceName = "Matthew";

    // The language to use when sending the message. For a list of supported
    // languages, see
    // https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
    static final String languageCode = "en-US";
```



```
// The content of the message. This example uses SSML to customize and
control
// certain aspects of the message, such as by adding pauses and changing
// phonation. The message can't contain any line breaks.
static final String ssmlMessage = "<speak>This is a test message sent from "
    + "<emphasis>Amazon Pinpoint</emphasis> "
    + "using the <break strength='weak'>AWS "
    + "SDK for Java. "
    + "<amazon:effect phonation='soft'>Thank "
    + "you for listening.</amazon:effect></speak>";

public static void main(String[] args) {

    final String usage = ""

        Usage:  <originationNumber> <destinationNumber>\s

        Where:
            originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
            destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s

        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String originationNumber = args[0];
    String destinationNumber = args[1];
    System.out.println("Sending a voice message");

    // Set the content type to application/json.
    List<String> listVal = new ArrayList<>();
    listVal.add("application/json");
    Map<String, List<String>> values = new HashMap<>();
    values.put("Content-Type", listVal);

    ClientOverrideConfiguration config2 =
ClientOverrideConfiguration.builder()
        .headers(values)
```

```
        .build();

        PinpointSmsVoiceClient client = PinpointSmsVoiceClient.builder()
            .overrideConfiguration(config2)
            .region(Region.US_EAST_1)
            .build();

        sendVoiceMsg(client, originationNumber, destinationNumber);
        client.close();
    }

    public static void sendVoiceMsg(PinpointSmsVoiceClient client, String
originationNumber,
        String destinationNumber) {
        try {
            SSMLMessageType ssmlMessageType = SSMLMessageType.builder()
                .languageCode(languageCode)
                .text(ssmlMessage)
                .voiceId(voiceName)
                .build();

            VoiceMessageContent content = VoiceMessageContent.builder()
                .ssmlMessage(ssmlMessageType)
                .build();

            SendVoiceMessageRequest voiceMessageRequest =
SendVoiceMessageRequest.builder()
                .destinationPhoneNumber(destinationNumber)
                .originationPhoneNumber(originationNumber)
                .content(content)
                .build();

            client.sendVoiceMessage(voiceMessageRequest);
            System.out.println("The message was sent successfully.");

        } catch (PinpointSmsVoiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [SendVoiceMessage](#)」の「」を参照してください。AWS SDK for Java 2.x

## SDK for Java 2.x を使用する Amazon Polly の例

次のコード例は、Amazon Polly AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- [アクション](#)

### アクション

#### DescribeVoices

次の例は、DescribeVoices を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「API リファレンス [DescribeVoices](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListLexicons

次の例は、ListLexicons を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
}
}
```

- API の詳細については、「API リファレンス [ListLexicons](#)」の「」を参照してください。AWS SDK for Java 2.x

## SynthesizeSpeech

次の例は、SynthesizeSpeech を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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;
}
```



```
}
```

- API の詳細については、「API リファレンス [SynthesizeSpeech](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## SDK for Java 2.x を使用する Amazon RDS の例

次のコード例は、Amazon RDS AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### 開始方法

#### Hello Amazon RDS

次のコード例は、Amazon RDS の使用を開始する方法を示しています。

#### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
```

```
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " + instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DescribeDBInstances](#)」を参照してください。

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### CreateDBInstance

次の例は、CreateDBInstance を使用する方法を説明しています。

#### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import com.google.gson.Gson;
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This example requires an AWS Secrets Manager secret that contains the
* database credentials. If you do not create a
* secret, this example will not work. For more details, see:
*
* https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating\_how-
services-use-secrets\_RS.html
*
*
*/
```

```
public class CreateDBInstance {
    public static long sleepTime = 20;

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbInstanceIdentifier> <dbName> <secretName>

            Where:
                dbInstanceIdentifier - The database instance identifier.\s
                dbName - The database name.\s
                secretName - The name of the AWS Secrets Manager secret that
contains the database credentials."
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String dbName = args[1];
        String secretName = args[2];
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();
```

```
        createDatabaseInstance(rdsClient, dbInstanceIdentifier, dbName,
user.getUsername(), user.getPassword());
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
            .region(region)

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

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void createDatabaseInstance(RdsClient rdsClient,
        String dbInstanceIdentifier,
        String dbName,
        String userName,
        String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .allocatedStorage(100)
                .dbName(dbName)
                .engine("mysql")
                .dbInstanceClass("db.m4.large")
                .engineVersion("8.0")
                .storageType("standard")
                .masterUsername(userName)
                .masterUserPassword(userPassword)
```

```
        .build());

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        // Loop until the cluster is ready.
        while (!instanceReady) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                instanceReadyStr = instance.dbInstanceStatus();
                if (instanceReadyStr.contains("available"))
                    instanceReady = true;
                else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
        System.out.println("Database instance is available!");
    } catch (RdsException | InterruptedException e) {
        System.err.println(e.getMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[CreateDBInstance](#)」を参照してください。

## CreateDBParameterGroup

次の例は、CreateDBParameterGroup を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [CreateDBParameterGroup](#)」を参照してください。AWS SDK for Java 2.x

## CreateDBSnapshot

次の例は、CreateDBSnapshot を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Create an Amazon RDS snapshot.
public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
    try {
        CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
        System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[CreateDBSnapshot](#)」を参照してください。



## DeleteDBInstance

次の例は、DeleteDBInstance を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDBInstance {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <dbInstanceIdentifier>\s

                Where:
                dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
```

```
Region region = Region.US_WEST_2;
RdsClient rdsClient = RdsClient.builder()
    .region(region)
    .build();

deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
rdsClient.close();
}

public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .deleteAutomatedBackups(true)
            .skipFinalSnapshot(true)
            .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DeleteDBInstance](#)」を参照してください。

## DeleteDBParameterGroup

次の例は、DeleteDBParameterGroup を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Delete the parameter group after database has been deleted.
// An exception is thrown if you attempt to delete the para group while database
// exists.
public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
    throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(dbARN) == 0) {
                    System.out.println(dbARN + " still exists");
                    didFind = true;
                }
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.

                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }
}
```

```
        // Delete the para group.
        DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
        .dbParameterGroupName(dbGroupName)
        .build();

        rdsClient.deleteDBParameterGroup(parameterGroupRequest);
        System.out.println(dbGroupName + " was deleted.");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DeleteDBParameterGroup](#)」を参照してください。AWS SDK for Java 2.x

## DescribeAccountAttributes

次の例は、DescribeAccountAttributes を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.AccountQuota;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeAccountAttributesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DescribeAccountAttributes {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        getAccountAttributes(rdsClient);
        rdsClient.close();
    }

    public static void getAccountAttributes(RdsClient rdsClient) {
        try {
            DescribeAccountAttributesResponse response =
rdsClient.describeAccountAttributes();
            List<AccountQuota> quotasList = response.accountQuotas();
            for (AccountQuota quotas : quotasList) {
                System.out.println("Name is: " + quotas.accountQuotaName());
                System.out.println("Max value is " + quotas.max());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- APIの詳細については、「APIリファレンス[DescribeAccountAttributes](#)」の「」を参照してください。AWS SDK for Java 2.x

## DescribeDBEngineVersions

次の例は、DescribeDBEngineVersions を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .defaultOnly(true)
            .engine("mysql")
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engineOb : engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "
                + engineOb.dbParameterGroupFamily());
            System.out.println("The name of the database engine " +
engineOb.engine());
            System.out.println("The version number of the database engine " +
engineOb.engineVersion());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API [DescribeDBEngineVersions](#) AWS SDK for Java 2.x」を参照してください。

## DescribeDBInstances

次の例は、DescribeDBInstances を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
        }
    }
}
```

```
        for (DBInstance instance : instanceList) {
            System.out.println("Instance ARN is: " + instance.dbInstanceArn());
            System.out.println("The Engine is " + instance.engine());
            System.out.println("Connection endpoint is" +
instance.endpoint().address());
        }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DescribeDBInstances](#)」を参照してください。

## DescribeDBParameterGroups

次の例は、DescribeDBParameterGroups を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
    try {
        DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .maxRecords(20)
            .build();

        DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
```



```
List<DBParameterGroup> groups = response.dbParameterGroups();
for (DBParameterGroup group : groups) {
    System.out.println("The group name is " +
group.dbParameterGroupName());
    System.out.println("The group description is " +
group.description());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API [DescribeDBParameterGroups](#) AWS SDK for Java 2.x」を参照してください。

## DescribeDBParameters

次の例は、DescribeDBParameters を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
    try {
        DescribeDbParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
```

```
        .dbParameterGroupName(dbGroupName)
        .source("user")
        .build();
    }

    DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
    List<Parameter> dbParameters = response.parameters();
    String paraName;
    for (Parameter para : dbParameters) {
        // Only print out information about either auto_increment_offset or
        // auto_increment_increment.
        paraName = para.parameterName();
        if ((paraName.compareTo("auto_increment_offset") == 0)
            || (paraName.compareTo("auto_increment_increment ") == 0)) {
            System.out.println("*** The parameter name is " + paraName);
            System.out.println("*** The parameter value is " +
para.parameterValue());
            System.out.println("*** The parameter data type is " +
para.dataType());
            System.out.println("*** The parameter description is " +
para.description());
            System.out.println("*** The parameter allowed values is " +
para.allowedValues());
        }
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DescribeDBParameters](#)」を参照してください。

## DescribeOrderableDBInstanceOptions

次の例は、DescribeOrderableDBInstanceOptions を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

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

- API の詳細については、「API リファレンス」の [DescribeOrderable「DBInstanceOptions」](#) を参照してください。 AWS SDK for Java 2.x

## GenerateRDSAuthToken

次の例は、GenerateRDSAuthToken を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

[RdsUtilities](#) クラスを使用して認証トークンを生成します。

```
public class GenerateRDSAuthToken {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbInstanceIdentifier> <masterUsername>

            Where:
                dbInstanceIdentifier - The database instance identifier.\s
                masterUsername - The master user name.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUsername = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        String token = getAuthToken(rdsClient, dbInstanceIdentifier,
masterUsername);
        System.out.println("The token response is " + token);
    }

    public static String getAuthToken(RdsClient rdsClient, String
dbInstanceIdentifier, String masterUsername) {

        RdsUtilities utilities = rdsClient.utilities();
```

```
    try {
        GenerateAuthenticationTokenRequest tokenRequest =
GenerateAuthenticationTokenRequest.builder()
            .credentialsProvider(ProfileCredentialsProvider.create())
            .username(masterUsername)
            .port(3306)
            .hostname(dbInstanceIdentifier)
            .build();

        return utilities.generateAuthenticationToken(tokenRequest);

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
}
```

- API の詳細については、「API リファレンス [GenerateRDSAuthToken](#)」を参照してください。  
AWS SDK for Java 2.x

## ModifyDBInstance

次の例は、ModifyDBInstance を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ModifyDBInstance {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbInstanceIdentifier> <dbSnapshotIdentifier>\s
            Where:
                dbInstanceIdentifier - The database instance identifier.\s
                masterUserPassword - The updated password that corresponds to
the master user name.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUserPassword = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        updateIntance(rdsClient, dbInstanceIdentifier, masterUserPassword);
        rdsClient.close();
    }

    public static void updateIntance(RdsClient rdsClient, String
dbInstanceIdentifier, String masterUserPassword) {
        try {
            // For a demo - modify the DB instance by modifying the master password.
            ModifyDbInstanceRequest modifyDbInstanceRequest =
ModifyDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .publiclyAccessible(true)

```

```
        .masterUserPassword(masterUserPassword)
        .build();

    ModifyDbInstanceResponse instanceResponse =
rdsClient.modifyDBInstance(modifyDbInstanceRequest);
    System.out.print("The ARN of the modified database is: " +
instanceResponse.dbInstance().dbInstanceArn());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、AWS SDK for Java 2.x API リファレンスの「[ModifyDBInstance](#)」を参照してください。

## ModifyDBParameterGroup

次の例は、ModifyDBParameterGroup を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();

        List<Parameter> paraList = new ArrayList<>();
```

```
        paraList.add(parameter1);
        ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .parameters(paraList)
            .build();

        ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
        System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [ModifyDBParameterGroup](#)」を参照してください。AWS SDK for Java 2.x

## RebootDBInstance

次の例は、RebootDBInstance を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
```



```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class RebootDBInstance {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbInstanceIdentifier>\s

            Where:
                dbInstanceIdentifier - The database instance identifier\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        rebootInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void rebootInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            RebootDbInstanceRequest rebootDbInstanceRequest =
RebootDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .build();

            RebootDbInstanceResponse instanceResponse =
rdsClient.rebootDBInstance(rebootDbInstanceRequest);
```

```
        System.out.print("The database " +
instanceResponse.dbInstance().dbInstanceArn() + " was rebooted");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[RebootDBInstance](#)」を参照してください。

## シナリオ

### DB インスタンスの使用を開始する

次のコードサンプルは、以下の操作方法を示しています。

- カスタム DB パラメータグループを作成し、パラメータ値を設定します。
- パラメータグループを使用するように設定した DB インスタンスを作成します。DB インスタンスにはデータベースも含まれています。
- インスタンスのスナップショットを取得します。
- インスタンスとパラメータグループを削除します。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

複数のオペレーションを実行します。

```
import com.google.gson.Gson;
import
software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotRequest;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotResponse;
import software.amazon.awssdk.services.rds.model.DBEngineVersion;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.DBParameterGroup;
import software.amazon.awssdk.services.rds.model.DBSnapshot;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsResponse;
import
    software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsResponse;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.OrderableDBInstanceOption;
import software.amazon.awssdk.services.rds.model.Parameter;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupRequest;
import
    software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbParameterGroupRequest;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *

```

```

* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This example requires an AWS Secrets Manager secret that contains the
* database credentials. If you do not create a
* secret, this example will not work. For details, see:
*
* https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating\_how-
services-use-secrets\_RS.html
*
* This Java example performs these tasks:
*
* 1. Returns a list of the available DB engines.
* 2. Selects an engine family and create a custom DB parameter group.
* 3. Gets the parameter groups.
* 4. Gets parameters in the group.
* 5. Modifies the auto_increment_offset parameter.
* 6. Gets and displays the updated parameters.
* 7. Gets a list of allowed engine versions.
* 8. Gets a list of micro instance classes available for the selected engine.
* 9. Creates an RDS database instance that contains a MySQL database and uses
* the parameter group.
* 10. Waits for the DB instance to be ready and prints out the connection
* endpoint value.
* 11. Creates a snapshot of the DB instance.
* 12. Waits for an RDS DB snapshot to be ready.
* 13. Deletes the RDS DB instance.
* 14. Deletes the parameter group.
*/
public class RDSScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

                Usage:
                <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
<dbName> <dbSnapshotIdentifier> <secretName>

                Where:
                dbGroupName - The database group name.\s

```

```
        dbParameterGroupFamily - The database parameter group name (for
example, mysql8.0).
        dbInstanceIdentifier - The database instance identifier\s
        dbName - The database name.\s
        dbSnapshotIdentifier - The snapshot identifier.\s
        secretName - The name of the AWS Secrets Manager secret that
contains the database credentials"
        """;

    if (args.length != 6) {
        System.out.println(usage);
        System.exit(1);
    }

    String dbGroupName = args[0];
    String dbParameterGroupFamily = args[1];
    String dbInstanceIdentifier = args[2];
    String dbName = args[3];
    String dbSnapshotIdentifier = args[4];
    String secretName = args[5];

    Gson gson = new Gson();
    User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
    String masterUsername = user.getUsername();
    String masterUserPassword = user.getPassword();

    Region region = Region.US_WEST_2;
    RdsClient rdsClient = RdsClient.builder()
        .region(region)
        .build();
    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon RDS example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Return a list of the available DB engines");
    describeDBEngines(rdsClient);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Create a custom parameter group");
    createDBParameterGroup(rdsClient, dbGroupName, dbParameterGroupFamily);
    System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("3. Get the parameter group");
describeDbParameterGroups(rdsClient, dbGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get the parameters in the group");
describeDbParameters(rdsClient, dbGroupName, 0);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Modify the auto_increment_offset parameter");
modifyDBParas(rdsClient, dbGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Display the updated value");
describeDbParameters(rdsClient, dbGroupName, -1);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of allowed engine versions");
getAllowedEngines(rdsClient, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Get a list of micro instance classes available for
the selected engine");
getMicroInstances(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(
    "9. Create an RDS database instance that contains a MySQL database
and uses the parameter group");
String dbARN = createDatabaseInstance(rdsClient, dbGroupName,
dbInstanceIdentifier, dbName, masterUsername,
    masterUserPassword);
System.out.println("The ARN of the new database is " + dbARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Wait for DB instance to be ready");
```

```
    waitForInstanceReady(rdsClient, dbInstanceIdentifier);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("11. Create a snapshot of the DB instance");
    createSnapshot(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("12. Wait for DB snapshot to be ready");
    waitForSnapshotReady(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("13. Delete the DB instance");
    deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("14. Delete the parameter group");
    deleteParaGroup(rdsClient, dbGroupName, dbARN);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The Scenario has successfully completed.");
    System.out.println(DASHES);

    rdsClient.close();
}

private static SecretsManagerClient getSecretClient() {
    Region region = Region.US_WEST_2;
    return SecretsManagerClient.builder()
        .region(region)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();
}

public static String getSecretValues(String secretName) {
    SecretsManagerClient secretClient = getSecretClient();
    GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
        .secretId(secretName)
        .build();
```

```
        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    // Delete the parameter group after database has been deleted.
    // An exception is thrown if you attempt to delete the para group while database
    // exists.
    public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
        throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(dbARN) == 0) {
                    System.out.println(dbARN + " still exists");
                    didFind = true;
                }
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.

                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }

    // Delete the para group.
    DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
```



```
        .dbParameterGroupName(dbGroupName)
        .build();

        rdsClient.deleteDBParameterGroup(parameterGroupRequest);
        System.out.println(dbGroupName + " was deleted.");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Delete the DB instance.
public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .deleteAutomatedBackups(true)
            .skipFinalSnapshot(true)
            .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the snapshot instance is available.
public static void waitForSnapshotReady(RdsClient rdsClient, String
dbInstanceIdentifier,
    String dbSnapshotIdentifier) {
    try {
        boolean snapshotReady = false;
        String snapshotReadyStr;
        System.out.println("Waiting for the snapshot to become available.");
```

```
        DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
        .dbSnapshotIdentifier(dbSnapshotIdentifier)
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .build();

        while (!snapshotReady) {
            DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
            List<DBSnapshot> snapshotList = response.dbSnapshots();
            for (DBSnapshot snapshot : snapshotList) {
                snapshotReadyStr = snapshot.status();
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }

        System.out.println("The Snapshot is available!");
    } catch (RdsException | InterruptedException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Create an Amazon RDS snapshot.
public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
    try {
        CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .dbSnapshotIdentifier(dbSnapshotIdentifier)
        .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
        System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

    } catch (RdsException e) {
```

```
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        String endpoint = "";
        while (!instanceReady) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                instanceReadyStr = instance.dbInstanceStatus();
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint().address();
                    instanceReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
        System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

    } catch (RdsException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Create a database instance and return the ARN of the database.
public static String createDatabaseInstance(RdsClient rdsClient,
```

```
        String dbGroupName,
        String dbInstanceIdentifier,
        String dbName,
        String masterUsername,
        String masterUserPassword) {

    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .allocatedStorage(100)
            .dbName(dbName)
            .dbParameterGroupName(dbGroupName)
            .engine("mysql")
            .dbInstanceClass("db.m4.large")
            .engineVersion("8.0")
            .storageType("standard")
            .masterUsername(masterUsername)
            .masterUserPassword(masterUserPassword)
            .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.println("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }

    return "";
}

// Get a list of micro instances.
public static void getMicroInstances(RdsClient rdsClient) {
    try {
        DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest
            .builder()
            .engine("mysql")
            .build();
```

```
DescribeOrderableDbInstanceOptionsResponse response = rdsClient
    .describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
List<OrderableDBInstanceOption> orderableDBInstances =
response.orderableDBInstanceOptions();
    for (OrderableDBInstanceOption dbInstanceOption : orderableDBInstances)
{
    System.out.println("The engine version is " +
dbInstanceOption.engineVersion());
    System.out.println("The engine description is " +
dbInstanceOption.engine());
    }

    } catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
    }
}

// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .engine("mysql")
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
        List<DBEngineVersion> dbEngines = response.dbEngineVersions();
        for (DBEngineVersion dbEngine : dbEngines) {
            System.out.println("The engine version is " +
dbEngine.engineVersion());
            System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();

        List<Parameter> paraList = new ArrayList<>();
        paraList.add(parameter1);
        ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .parameters(paraList)
            .build();

        ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
        System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
    try {
        DescribeDbParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .source("user")
                .build();
        }
    }
}
```

```
DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
List<Parameter> dbParameters = response.parameters();
String paraName;
for (Parameter para : dbParameters) {
    // Only print out information about either auto_increment_offset or
    // auto_increment_increment.
    paraName = para.parameterName();
    if ((paraName.compareTo("auto_increment_offset") == 0)
        || (paraName.compareTo("auto_increment_increment ") == 0)) {
        System.out.println("*** The parameter name is " + paraName);
        System.out.println("*** The parameter value is " +
para.parameterValue());
        System.out.println("*** The parameter data type is " +
para.dataType());
        System.out.println("*** The parameter description is " +
para.description());
        System.out.println("*** The parameter allowed values is " +
para.allowedValues());
    }
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
    try {
        DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .maxRecords(20)
            .build();

        DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
        List<DBParameterGroup> groups = response.dbParameterGroups();
        for (DBParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbParameterGroupName());
        }
    }
}
```

```
        System.out.println("The group description is " +
group.description());
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .defaultOnly(true)
            .engine("mysql")
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();
    }
}
```



```
// Get all DBEngineVersion objects.
for (DBEngineVersion engineOb : engines) {
    System.out.println("The name of the DB parameter group family for
the database engine is "
        + engineOb.dbParameterGroupFamily());
    System.out.println("The name of the database engine " +
engineOb.engine());
    System.out.println("The version number of the database engine " +
engineOb.engineVersion());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [CreateDBInstance](#)
  - [CreateDBParameterGroup](#)
  - [CreateDBSnapshot](#)
  - [DeleteDBInstance](#)
  - [DeleteDBParameterGroup](#)
  - [DescribeDBEngineVersions](#)
  - [DescribeDBInstances](#)
  - [DescribeDBParameterGroups](#)
  - [DescribeDBParameters](#)
  - [DescribeDBSnapshots](#)
  - [DescribeOrderableDBInstanceOptions](#)
  - [ModifyDBParameterGroup](#)

## SDK for Java 2.x を使用した Amazon Redshift の例

次のコード例は、Amazon Redshift AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### 開始方法

#### Hello Amazon Redshift

以下のコード例は、Amazon Redshift の使用を開始する方法を示しています。

#### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

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

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[describeClusters](#)」を参照してください。

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### CreateCluster

次の例は、CreateCluster を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

クラスターを作成する。

```
public static void createCluster(RedshiftClient redshiftClient, String
clusterId, String masterUsername,
                                String masterUserPassword) {
    try {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(masterUsername)
            .masterUserPassword(masterUserPassword)
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        CreateClusterResponse clusterResponse =
redshiftClient.createCluster(clusterRequest);
        System.out.println("Created cluster " +
clusterResponse.cluster().clusterIdentifier());

    } catch (RedshiftException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [CreateCluster](#)」の「」を参照してください。  
AWS SDK for Java 2.x

**CreateTable**

次の例は、CreateTable を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void createTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName) {
    try {
        ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
        .clusterIdentifier(clusterId)
        .dbUser(userName)
        .database(databaseName)
        .sql("CREATE TABLE Movies ("
            + "id INT PRIMARY KEY, "
            + "title VARCHAR(100), "
            + "year INT)")
        .build();

        redshiftDataClient.executeStatement(createTableRequest);
        System.out.println("Table created: Movies");

    } catch (RedshiftDataException e) {
        System.err.println("Error creating table: " + e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [CreateTable](#)」の「」を参照してください。 AWS SDK for Java 2.x

**DeleteCluster**

次の例は、DeleteCluster を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

クラスターを削除します。

```
public static void deleteRedshiftCluster(RedshiftClient redshiftClient, String
clusterId) {
    try {
        DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
        .clusterIdentifier(clusterId)
        .skipFinalClusterSnapshot(true)
        .build();

        DeleteClusterResponse response =
redshiftClient.deleteCluster(deleteClusterRequest);
        System.out.println("The status is " +
response.cluster().clusterStatus());

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DeleteCluster](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DescribeClusters

次の例は、DescribeClusters を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## クラスターを記述する

```
public static void waitForClusterReady(RedshiftClient redshiftClient, String
clusterId) {
    boolean clusterReady = false;
    String clusterReadyStr;
    System.out.println("Waiting for cluster to become available. This may take a
few mins.");
    try {
        DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();
        long startTime = System.currentTimeMillis();

        // Loop until the cluster is ready.
        while (!clusterReady) {
            DescribeClustersResponse clusterResponse =
redshiftClient.describeClusters(clustersRequest);
            List<Cluster> clusterList = clusterResponse.clusters();
            for (Cluster cluster : clusterList) {
                clusterReadyStr = cluster.clusterStatus();
                if (clusterReadyStr.contains("available"))
                    clusterReady = true;
                else {
                    long elapsedTimeMillis = System.currentTimeMillis() -
startTime;

                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;

                    System.out.printf("Elapsed Time: %02d:%02d - Waiting for
cluster... %n", minutes, seconds);
                    TimeUnit.SECONDS.sleep(5);
                }
            }
        }
    }
}
```

```
    }  
  }  
  
  long elapsedTimeMillis = System.currentTimeMillis() - startTime;  
  long elapsedSeconds = elapsedTimeMillis / 1000;  
  long minutes = elapsedSeconds / 60;  
  long seconds = elapsedSeconds % 60;  
  
  System.out.println(String.format("Cluster is available! Total Elapsed  
Time: %02d:%02d", minutes, seconds));  
  
  } catch (RedshiftException | InterruptedException e) {  
    System.err.println(e.getMessage());  
    System.exit(1);  
  }  
}
```

- API の詳細については、「API リファレンス [DescribeClusters](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DescribeStatement

次の例は、DescribeStatement を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void checkStatement(RedshiftDataClient redshiftDataClient, String  
sqlId) {  
  try {  
    DescribeStatementRequest statementRequest =  
DescribeStatementRequest.builder()  
      .id(sqlId)  
      .build();
```



```
String status;
while (true) {
    DescribeStatementResponse response =
redshiftDataClient.describeStatement(statementRequest);
    status = response.statusAsString();
    System.out.println("..." + status);

    if (status.compareTo("FAILED") == 0 ) {
        System.out.println("The Query Failed. Ending program");
        System.exit(1);

    } else if (status.compareTo("FINISHED") == 0) {
        break;
    }
    TimeUnit.SECONDS.sleep(1);
}

System.out.println("The statement is finished!");

} catch (RedshiftDataException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス [DescribeStatement](#)」の「」を参照してください。AWS SDK for Java 2.x

## GetStatementResult

次の例は、GetStatementResult を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

ステートメントの結果を確認します。

```
public static void getResults(RedshiftDataClient redshiftDataClient, String
statementId) {
    try {
        GetStatementResultRequest resultRequest =
        GetStatementResultRequest.builder()
            .id(statementId)
            .build();

        // Extract and print the field values using streams.
        GetStatementResultResponse response =
        redshiftDataClient.getStatementResult(resultRequest);
        response.records().stream()
            .flatMap(List::stream)
            .map(Field::stringValue)
            .filter(value -> value != null)
            .forEach(value -> System.out.println("The Movie title field is " +
value));

    } catch (RedshiftDataException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [GetStatementResult](#)」の「」を参照してください。AWS SDK for Java 2.x

## Insert

次の例は、Insert を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void popTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName, String fileName, int number)
throws IOException {
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {
        if (t == number)
            break;
        currentNode = (ObjectNode) iter.next();
        int year = currentNode.get("year").asInt();
        String title = currentNode.get("title").asText();

        // Use SqlParameter to avoid SQL injection.
        List<SqlParameter> parameterList = new ArrayList<>();
        String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";

        // Create the parameters.
        SqlParameter idParam = SqlParameter.builder()
            .name("id")
            .value(String.valueOf(t))
            .build();

        SqlParameter titleParam= SqlParameter.builder()
            .name("title")
            .value(title)
            .build();

        SqlParameter yearParam = SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();
        parameterList.add(idParam);
        parameterList.add(titleParam);
        parameterList.add(yearParam);

        try {
            ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
```

```
        .clusterIdentifier(clusterId)
        .sql(sqlStatement)
        .database(databaseName)
        .dbUser(userName)
        .parameters(parameterList)
        .build();

        redshiftDataClient.executeStatement(insertStatementRequest);
        System.out.println("Inserted: " + title + " (" + year + ")");
        t++;

    } catch (RedshiftDataException e) {
        System.err.println("Error inserting data: " + e.getMessage());
        System.exit(1);
    }
}
System.out.println(t + " records were added to the Movies table. ");
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[Insert](#)」を参照してください。

## ModifyCluster

次の例は、ModifyCluster を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

クラスターを変更します。

```
public static void modifyCluster(RedshiftClient redshiftClient, String
clusterId) {
    try {
        ModifyClusterRequest modifyClusterRequest =
        ModifyClusterRequest.builder()
```

```
        .clusterIdentifier(clusterId)
        .preferredMaintenanceWindow("wed:07:30-wed:08:00")
        .build();

        ModifyClusterResponse clusterResponse =
redshiftClient.modifyCluster(modifyClusterRequest);
        System.out.println("The modified cluster was successfully modified and
has "
            + clusterResponse.cluster().preferredMaintenanceWindow() + " as the
maintenance window");

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [ModifyCluster](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## Query

次の例は、Query を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例  
を見つけて、設定と実行の方法を確認してください。

テーブルに対してクエリを実行します。

```
public static String queryMoviesByYear(RedshiftDataClient redshiftDataClient,
                                       String database,
                                       String dbUser,
                                       int year,
                                       String clusterId) {
```

```
try {
    String sqlStatement = " SELECT * FROM Movies WHERE year = :year";
    SqlParameter yearParam= SqlParameter.builder()
        .name("year")
        .value(String.valueOf(year))
        .build();

    ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
        .clusterIdentifier(clusterId)
        .database(database)
        .dbUser(dbUser)
        .parameters(yearParam)
        .sql(sqlStatement)
        .build();

    ExecuteStatementResponse response =
redshiftDataClient.executeStatement(statementRequest);
    return response.id();

} catch (RedshiftDataException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[Query](#)」を参照してください。

## シナリオ

### Amazon Redshift の使用開始

次のコード例は、Amazon Redshift のテーブル、項目、クエリを操作する方法を示しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.ObjectNode;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import software.amazon.awssdk.services.redshift.model.Cluster;
import software.amazon.awssdk.services.redshift.model.CreateClusterRequest;
import software.amazon.awssdk.services.redshift.model.CreateClusterResponse;
import software.amazon.awssdk.services.redshift.model.DeleteClusterRequest;
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;
import software.amazon.awssdk.services.redshift.model.DescribeClustersRequest;
import software.amazon.awssdk.services.redshift.model.DescribeClustersResponse;
import software.amazon.awssdk.services.redshift.model.ModifyClusterRequest;
import software.amazon.awssdk.services.redshift.model.ModifyClusterResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import software.amazon.awssdk.services.redshiftdata.RedshiftDataClient;
import software.amazon.awssdk.services.redshiftdata.model.DescribeStatementRequest;
import software.amazon.awssdk.services.redshiftdata.model.DescribeStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.Field;
import software.amazon.awssdk.services.redshiftdata.model.GetStatementResultRequest;
import
    software.amazon.awssdk.services.redshiftdata.model.GetStatementResultResponse;
import software.amazon.awssdk.services.redshiftdata.model.ListDatabasesRequest;
import software.amazon.awssdk.services.redshiftdata.model.RedshiftDataException;
import software.amazon.awssdk.services.redshiftdata.model.SqlParameter;
import
    software.amazon.awssdk.services.redshiftdata.paginators.ListDatabasesIterable;
import com.fasterxml.jackson.core.JsonParser;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
```

```
import java.util.Iterator;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Prompts the user for a unique cluster ID or use the default value.
 * 2. Creates a Redshift cluster with the specified or default cluster Id value.
 * 3. Waits until the Redshift cluster is available for use.
 * 4. Lists all databases using a pagination API call.
 * 5. Creates a table named "Movies" with fields ID, title, and year.
 * 6. Inserts a specified number of records into the "Movies" table by reading the
 * Movies JSON file.
 * 7. Prompts the user for a movie release year.
 * 8. Runs a SQL query to retrieve movies released in the specified year.
 * 9. Modifies the Redshift cluster.
 * 10. Prompts the user for confirmation to delete the Redshift cluster.
 * 11. If confirmed, deletes the specified Redshift cluster.
 */

public class RedshiftScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws Exception {
        final String usage = ""

            Usage:
                <jsonFilePath>\s

            Where:
                jsonFilePath - The path to the Movies JSON file (you can locate that
file in ../../../../resources/sample_files/movies.json)
                """;

        if (args.length != 1) {
```



```
        System.out.println(usage);
        System.exit(1);
    }

    String jsonFilePath = args[0];
    String userName;
    String userPassword;
    String databaseName = "dev" ;
    Scanner scanner = new Scanner(System.in);

    Region region = Region.US_EAST_1;
    RedshiftClient redshiftClient = RedshiftClient.builder()
        .region(region)
        .build();

    RedshiftDataClient redshiftDataClient = RedshiftDataClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon Redshift SDK Getting Started
scenario.");
    System.out.println(""""
    This Java program demonstrates how to interact with Amazon Redshift by using
the AWS SDK for Java (v2).\s
    Amazon Redshift is a fully managed, petabyte-scale data warehouse service
hosted in the cloud.

    The program's primary functionalities include cluster creation, verification
of cluster readiness,\s
    list databases, table creation, data population within the table, and
execution of SQL statements.
    Furthermore, it demonstrates the process of querying data from the Movie
table.\s

    Upon completion of the program, all AWS resources are cleaned up.
    """);

    System.out.println("Lets get started...");
    System.out.println("Please enter your user name (default is awsuser)");
    String user = scanner.nextLine();
    userName = user.isEmpty() ? "awsuser" : user;
    System.out.println(DASHES);
```

```
System.out.println("Please enter your user password (default is
AwsUser1000)");
String userpass = scanner.nextLine();
userPassword = userpass.isEmpty() ? "AwsUser1000" : userpass;
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("A Redshift cluster refers to the collection of computing
resources and storage that work together to process and analyze large volumes of
data.");
System.out.println("Enter a cluster id value (default is redshift-cluster-
movies): ");
String userClusterId = scanner.nextLine();
String clusterId = userClusterId.isEmpty() ? "redshift-cluster-movies" :
userClusterId;
createCluster(redshiftClient, clusterId, userName, userPassword);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Wait until "+clusterId+" is available.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
waitForClusterReady(redshiftClient, clusterId);
System.out.println(DASHES);

System.out.println(DASHES);
String databaseInfo = ""
    When you created $clusteridD, the dev database is created by default and
used in this scenario.\s

    To create a custom database, you need to have a CREATEDB privilege.\s
    For more information, see the documentation here: https://
docs.aws.amazon.com/redshift/latest/dg/r\_CREATE\_DATABASE.html.
"".replace("$clusteridD", clusterId);

System.out.println(databaseInfo);
System.out.print("Press Enter to continue...");
scanner.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("List databases in "+clusterId);
System.out.print("Press Enter to continue...");
scanner.nextLine();
```

```
listAllDatabases(redshiftDataClient, clusterId, userName, databaseName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Now you will create a table named Movies.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
createTable(redshiftDataClient, clusterId, databaseName, userName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Populate the Movies table using the Movies.json file.");
System.out.println("Specify the number of records you would like to add to
the Movies Table.");
System.out.println("Please enter a value between 50 and 200.");
int numRecords;
do {
    System.out.print("Enter a value: ");
    while (!scanner.hasNextInt()) {
        System.out.println("Invalid input. Please enter a value between 50
and 200.");
        System.out.print("Enter a year: ");
        scanner.next();
    }
    numRecords = scanner.nextInt();
} while (numRecords < 50 || numRecords > 200);
popTable(redshiftDataClient, clusterId, databaseName, userName,
jsonFilePath, numRecords);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Query the Movies table by year. Enter a value between
2012-2014.");
int movieYear;
do {
    System.out.print("Enter a year: ");
    while (!scanner.hasNextInt()) {
        System.out.println("Invalid input. Please enter a valid year between
2012 and 2014.");
        System.out.print("Enter a year: ");
        scanner.next();
    }
    movieYear = scanner.nextInt();
    scanner.nextLine();
}
```

```
    } while (movieYear < 2012 || movieYear > 2014);

    String id = queryMoviesByYear(redshiftDataClient, databaseName, userName,
movieYear, clusterId);
    System.out.println("The identifier of the statement is " + id);
    checkStatement(redshiftDataClient, id);
    getResults(redshiftDataClient, id);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Now you will modify the Redshift cluster.");
    System.out.print("Press Enter to continue...");
    scanner.nextLine();
    modifyCluster(redshiftClient, clusterId);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Would you like to delete the Amazon Redshift cluster?
(y/n)");
    String delAns = scanner.nextLine().trim();
    if (delAns.equalsIgnoreCase("y")) {
        System.out.println("You selected to delete " +clusterId);
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        deleteRedshiftCluster(redshiftClient, clusterId);
    } else {
        System.out.println("The "+clusterId +" was not deleted");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("This concludes the Amazon Redshift SDK Getting Started
scenario.");
    System.out.println(DASHES);
}

public static void listAllDatabases(RedshiftDataClient redshiftDataClient,
String clusterId, String dbUser, String database) {
    try {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(dbUser)
            .database(database)
            .build();
```

```
        ListDatabasesIterable listDatabasesIterable =
redshiftDataClient.listDatabasesPaginator(databasesRequest);
        listDatabasesIterable.stream()
            .flatMap(r -> r.databases().stream())
            .forEach(db -> System.out
                .println("The database name is : " + db));

    } catch (RedshiftDataException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteRedshiftCluster(RedshiftClient redshiftClient, String
clusterId) {
    try {
        DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .skipFinalClusterSnapshot(true)
            .build();

        DeleteClusterResponse response =
redshiftClient.deleteCluster(deleteClusterRequest);
        System.out.println("The status is " +
response.cluster().clusterStatus());

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void popTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName, String fileName, int number)
throws IOException {
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {
```

```
        if (t == number)
            break;
        currentNode = (ObjectNode) iter.next();
        int year = currentNode.get("year").asInt();
        String title = currentNode.get("title").asText();

        // Use SqlParameter to avoid SQL injection.
        List<SqlParameter> parameterList = new ArrayList<>();
        String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";

        // Create the parameters.
        SqlParameter idParam = SqlParameter.builder()
            .name("id")
            .value(String.valueOf(t))
            .build();

        SqlParameter titleParam= SqlParameter.builder()
            .name("title")
            .value(title)
            .build();

        SqlParameter yearParam = SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();
        parameterList.add(idParam);
        parameterList.add(titleParam);
        parameterList.add(yearParam);

        try {
            ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                .clusterIdentifier(clusterId)
                .sql(sqlStatement)
                .database(databaseName)
                .dbUser(userName)
                .parameters(parameterList)
                .build();

            redshiftDataClient.executeStatement(insertStatementRequest);
            System.out.println("Inserted: " + title + " (" + year + ")");
            t++;
        }
```

```
        } catch (RedshiftDataException e) {
            System.err.println("Error inserting data: " + e.getMessage());
            System.exit(1);
        }
    }
    System.out.println(t + " records were added to the Movies table. ");
}

public static void checkStatement(RedshiftDataClient redshiftDataClient, String
sqlId) {
    try {
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();

        String status;
        while (true) {
            DescribeStatementResponse response =
redshiftDataClient.describeStatement(statementRequest);
            status = response.statusAsString();
            System.out.println("..." + status);

            if (status.compareTo("FAILED") == 0 ) {
                System.out.println("The Query Failed. Ending program");
                System.exit(1);
            }

            } else if (status.compareTo("FINISHED") == 0) {
                break;
            }
            TimeUnit.SECONDS.sleep(1);
        }

        System.out.println("The statement is finished!");
    } catch (RedshiftDataException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void modifyCluster(RedshiftClient redshiftClient, String
clusterId) {
    try {
```

```
        ModifyClusterRequest modifyClusterRequest =
ModifyClusterRequest.builder()
    .clusterIdentifier(clusterId)
    .preferredMaintenanceWindow("wed:07:30-wed:08:00")
    .build();

        ModifyClusterResponse clusterResponse =
redshiftClient.modifyCluster(modifyClusterRequest);
        System.out.println("The modified cluster was successfully modified and
has "
            + clusterResponse.cluster().preferredMaintenanceWindow() + " as the
maintenance window");

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String queryMoviesByYear(RedshiftDataClient redshiftDataClient,
                                        String database,
                                        String dbUser,
                                        int year,
                                        String clusterId) {

    try {
        String sqlStatement = " SELECT * FROM Movies WHERE year = :year";
        SqlParameter yearParam= SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();

        ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .database(database)
            .dbUser(dbUser)
            .parameters(yearParam)
            .sql(sqlStatement)
            .build();

        ExecuteStatementResponse response =
redshiftDataClient.executeStatement(statementRequest);
        return response.id();
    }
}
```



```
    } catch (RedshiftDataException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void getResults(RedshiftDataClient redshiftDataClient, String
statementId) {
    try {
        GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
            .id(statementId)
            .build();

        // Extract and print the field values using streams.
        GetStatementResultResponse response =
redshiftDataClient.getStatementResult(resultRequest);
        response.records().stream()
            .flatMap(List::stream)
            .map(Field::stringValue)
            .filter(value -> value != null)
            .forEach(value -> System.out.println("The Movie title field is " +
value));

    } catch (RedshiftDataException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void waitForClusterReady(RedshiftClient redshiftClient, String
clusterId) {
    boolean clusterReady = false;
    String clusterReadyStr;
    System.out.println("Waiting for cluster to become available. This may take a
few mins.");
    try {
        DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();
        long startTime = System.currentTimeMillis();
```

```
// Loop until the cluster is ready.
while (!clusterReady) {
    DescribeClustersResponse clusterResponse =
redshiftClient.describeClusters(clustersRequest);
    List<Cluster> clusterList = clusterResponse.clusters();
    for (Cluster cluster : clusterList) {
        clusterReadyStr = cluster.clusterStatus();
        if (clusterReadyStr.contains("available"))
            clusterReady = true;
        else {
            long elapsedTimeMillis = System.currentTimeMillis() -
startTime;

            long elapsedSeconds = elapsedTimeMillis / 1000;
            long minutes = elapsedSeconds / 60;
            long seconds = elapsedSeconds % 60;

            System.out.printf("Elapsed Time: %02d:%02d - Waiting for
cluster... %n", minutes, seconds);
            TimeUnit.SECONDS.sleep(5);
        }
    }
}

long elapsedTimeMillis = System.currentTimeMillis() - startTime;
long elapsedSeconds = elapsedTimeMillis / 1000;
long minutes = elapsedSeconds / 60;
long seconds = elapsedSeconds % 60;

System.out.println(String.format("Cluster is available! Total Elapsed
Time: %02d:%02d", minutes, seconds));

} catch (RedshiftException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void createTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName) {
    try {
        ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
```

```
        .dbUser(userName)
        .database(databaseName)
        .sql("CREATE TABLE Movies ("
            + "id INT PRIMARY KEY, "
            + "title VARCHAR(100), "
            + "year INT)")
        .build();

        redshiftDataClient.executeStatement(createTableRequest);
        System.out.println("Table created: Movies");

    } catch (RedshiftDataException e) {
        System.err.println("Error creating table: " + e.getMessage());
        System.exit(1);
    }
}

public static void createCluster(RedshiftClient redshiftClient, String
clusterId, String masterUsername,
                                String masterUserPassword) {
    try {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(masterUsername)
            .masterUserPassword(masterUserPassword)
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        CreateClusterResponse clusterResponse =
redshiftClient.createCluster(clusterRequest);
        System.out.println("Created cluster " +
clusterResponse.cluster().clusterIdentifier());

    } catch (RedshiftException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [createCluster](#)
  - [describeClusters](#)
  - [describeStatement](#)
  - [executeStatement](#)
  - [getStatementResult](#)
  - [listDatabasesPaginator](#)
  - [modifyCluster](#)

## SDK for Java 2.x を使用する Amazon Rekognition の例

次のコード例は、Amazon Rekognition AWS SDK for Java 2.x でを使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。Amazon Rekognition

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

### トピック

- [アクション](#)
- [シナリオ](#)

### アクション

#### CompareFaces

次の例は、CompareFaces を使用する方法を説明しています。

詳細については、「[イメージ内の顔を比較する](#)」を参照してください。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.CompareFacesRequest;
import software.amazon.awssdk.services.rekognition.model.CompareFacesResponse;
import software.amazon.awssdk.services.rekognition.model.CompareFacesMatch;
import software.amazon.awssdk.services.rekognition.model.ComparedFace;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CompareFaces {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <pathSource> <pathTarget>

                Where:
                pathSource - The path to the source image (for example, C:\\AWS\\
\\pic1.png).\\s
                pathTarget - The path to the target image (for example, C:\\AWS\\
\\pic2.png).\\s
    }
}
```

```
        """);

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    Float similarityThreshold = 70F;
    String sourceImage = args[0];
    String targetImage = args[1];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    compareTwoFaces(rekClient, similarityThreshold, sourceImage, targetImage);
    rekClient.close();
}

public static void compareTwoFaces(RekognitionClient rekClient, Float
similarityThreshold, String sourceImage,
    String targetImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        InputStream tarStream = new FileInputStream(targetImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        SdkBytes targetBytes = SdkBytes.fromInputStream(tarStream);

        // Create an Image object for the source image.
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        Image tarImage = Image.builder()
            .bytes(targetBytes)
            .build();

        CompareFacesRequest facesRequest = CompareFacesRequest.builder()
            .sourceImage(souImage)
            .targetImage(tarImage)
            .similarityThreshold(similarityThreshold)
            .build();

        // Compare the two images.
```

```
CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
List<CompareFacesMatch> faceDetails = compareFacesResult.faceMatches();
for (CompareFacesMatch match : faceDetails) {
    ComparedFace face = match.face();
    BoundingBox position = face.boundingBox();
    System.out.println("Face at " + position.left().toString()
        + " " + position.top()
        + " matches with " + face.confidence().toString()
        + "% confidence.");
}
List<ComparedFace> uncomparing = compareFacesResult.unmatchedFaces();
System.out.println("There was " + uncomparing.size() + " face(s) that did
not match");
System.out.println("Source image rotation: " +
compareFacesResult.sourceImageOrientationCorrection());
System.out.println("target image rotation: " +
compareFacesResult.targetImageOrientationCorrection());

} catch (RekognitionException | FileNotFoundException e) {
    System.out.println("Failed to load source image " + sourceImage);
    System.exit(1);
}
}
}
```

- API の詳細については、「API リファレンス [CompareFaces](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateCollection

次の例は、CreateCollection を使用する方法を説明しています。

詳細については、「[コレクションを作成する](#)」を参照してください。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCollection {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <collectionName>\s

            Where:
                collectionName - The name of the collection.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();
```



```
        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void createMyCollection(RecognitionClient 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 (RecognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [CreateCollection](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteCollection

次の例は、DeleteCollection を使用する方法を説明しています。

詳細については、「[コレクションを削除する](#)」を参照してください。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId>\s

                Where:
                    collectionId - The id of the collection to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
```

```
        System.out.println("Deleting collection: " + collectionId);
        deleteMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void deleteMyCollection(RekognitionClient rekClient, String
collectionId) {
        try {
            DeleteCollectionRequest deleteCollectionRequest =
DeleteCollectionRequest.builder()
                .collectionId(collectionId)
                .build();

            DeleteCollectionResponse deleteCollectionResponse =
rekClient.deleteCollection(deleteCollectionRequest);
            System.out.println(collectionId + ": " +
deleteCollectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [DeleteCollection](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteFaces

次の例は、DeleteFaces を使用する方法を説明しています。

詳細については、「[コレクションから顔を削除する](#)」を参照してください。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例  
を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteFacesFromCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <faceId>\s

                Where:
                    collectionId - The id of the collection from which faces are
deleted.\s

                    faceId - The id of the face to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }
}
```

```
public static void deleteFacesCollection(RekognitionClient rekClient,
    String collectionId,
    String faceId) {

    try {
        DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
            .collectionId(collectionId)
            .faceIds(faceId)
            .build();

        rekClient.deleteFaces(deleteFacesRequest);
        System.out.println("The face was deleted from the collection.");

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DeleteFaces](#)」の「」を参照してください。AWS SDK for Java 2.x

## DescribeCollection

次の例は、DescribeCollection を使用する方法を説明しています。

詳細については、「[コレクションを定義する](#)」を参照してください。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
```

```
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeCollection {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <collectionName>

            Where:
                collectionName - The name of the Amazon Rekognition collection.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    public static void describeColl(RekognitionClient rekClient, String
collectionName) {
        try {
            DescribeCollectionRequest describeCollectionRequest =
DescribeCollectionRequest.builder()
                .collectionId(collectionName)
                .build();

            DescribeCollectionResponse describeCollectionResponse = rekClient
```

```
        .describeCollection(describeCollectionRequest);
        System.out.println("Collection Arn : " +
describeCollectionResponse.collectionARN());
        System.out.println("Created : " +
describeCollectionResponse.creationTimestamp().toString());

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [DescribeCollection](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DetectFaces

次の例は、DetectFaces を使用する方法を説明しています。

詳細については、「[イメージ内の顔を検出する](#)」を参照してください。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.DetectFacesRequest;
import software.amazon.awssdk.services.rekognition.model.DetectFacesResponse;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceDetail;
import software.amazon.awssdk.services.rekognition.model.AgeRange;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
```

```
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectFaces {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectFacesinImage(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectFacesinImage(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
```



```
// Create an Image object for the source image.
Image souImage = Image.builder()
    .bytes(sourceBytes)
    .build();

DetectFacesRequest facesRequest = DetectFacesRequest.builder()
    .attributes(Attribute.ALL)
    .image(souImage)
    .build();

DetectFacesResponse facesResponse = rekClient.detectFaces(facesRequest);
List<FaceDetail> faceDetails = facesResponse.faceDetails();
for (FaceDetail face : faceDetails) {
    AgeRange ageRange = face.ageRange();
    System.out.println("The detected face is estimated to be between "
        + ageRange.low().toString() + " and " +
ageRange.high().toString()
        + " years old.");

    System.out.println("There is a smile : " +
face.smile().value().toString());
}

} catch (RekognitionException | FileNotFoundException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
}
```

- API の詳細については、「API リファレンス[DetectFaces](#)」の「」を参照してください。AWS SDK for Java 2.x

## DetectLabels

次の例は、DetectLabels を使用する方法を説明しています。

詳細については、「[イメージ内のラベルを検出する](#)」を参照してください。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsRequest;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectLabels {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <sourceImage>

                Where:
                    sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String sourceImage = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    detectImageLabels(rekClient, sourceImage);
    rekClient.close();
}

public static void detectImageLabels(RekognitionClient rekClient, String
sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Create an Image object for the source image.
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectLabelsRequest detectLabelsRequest = DetectLabelsRequest.builder()
            .image(souImage)
            .maxLabels(10)
            .build();

        DetectLabelsResponse labelsResponse =
rekClient.detectLabels(detectLabelsRequest);
        List<Label> labels = labelsResponse.labels();
        System.out.println("Detected labels for the given photo");
        for (Label label : labels) {
            System.out.println(label.name() + ": " +
label.confidence().toString());
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [DetectLabels](#)」の「」を参照してください。AWS SDK for Java 2.x

## DetectModerationLabels

次の例は、DetectModerationLabels を使用する方法を説明しています。

詳細については、「[不適切なイメージを検出する](#)」を参照してください。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.ModerationLabel;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class DetectModerationLabels {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
            """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectModLabels(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectModLabels(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
                .image(souImage)
                .minConfidence(60F)
                .build();

            DetectModerationLabelsResponse moderationLabelsResponse = rekClient
                .detectModerationLabels(moderationLabelsRequest);
```

```
        List<ModerationLabel> labels =
moderationLabelsResponse.moderationLabels();
        System.out.println("Detected labels for image");
        for (ModerationLabel label : labels) {
            System.out.println("Label: " + label.name()
                + "\n Confidence: " + label.confidence().toString() + "%"
                + "\n Parent:" + label.parentName());
        }

    } catch (RekognitionException | FileNotFoundException e) {
        e.printStackTrace();
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [DetectModerationLabels](#)」の「」を参照してください。AWS SDK for Java 2.x

## DetectText

次の例は、DetectText を使用する方法を説明しています。

詳細については、「[イメージ内のテキストを検出する](#)」を参照してください。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DetectTextRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectTextResponse;
import software.amazon.awssdk.services.rekognition.model.TextDetection;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
```

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectText {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image that contains text (for
example, C:\\AWS\\pic1.png).\\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectTextLabels(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectTextLabels(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
```

```
Image souImage = Image.builder()
    .bytes(sourceBytes)
    .build();

DetectTextRequest textRequest = DetectTextRequest.builder()
    .image(souImage)
    .build();

DetectTextResponse textResponse = rekClient.detectText(textRequest);
List<TextDetection> textCollection = textResponse.textDetections();
System.out.println("Detected lines and words");
for (TextDetection text : textCollection) {
    System.out.println("Detected: " + text.detectedText());
    System.out.println("Confidence: " + text.confidence().toString());
    System.out.println("Id : " + text.id());
    System.out.println("Parent Id: " + text.parentId());
    System.out.println("Type: " + text.type());
    System.out.println();
}

} catch (RekognitionException | FileNotFoundException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
}
```

- API の詳細については、「API リファレンス [DetectText](#)」の「」を参照してください。AWS SDK for Java 2.x

## IndexFaces

次の例は、IndexFaces を使用する方法を説明しています。

詳細については、「[コレクションに顔を追加する](#)」を参照してください。



## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.IndexFacesResponse;
import software.amazon.awssdk.services.rekognition.model.IndexFacesRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.QualityFilter;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceRecord;
import software.amazon.awssdk.services.rekognition.model.UnindexedFace;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Reason;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddFacesToCollection {
    public static void main(String[] args) {

        final String usage = ""

            Usage:      <collectionId> <sourceImage>

            Where:
                collectionName - The name of the collection.
```

```
        sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\s
        """";

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String collectionId = args[0];
    String sourceImage = args[1];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    addToCollection(rekClient, collectionId, sourceImage);
    rekClient.close();
}

public static void addToCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        IndexFacesRequest facesRequest = IndexFacesRequest.builder()
            .collectionId(collectionId)
            .image(souImage)
            .maxFaces(1)
            .qualityFilter(QualityFilter.AUTO)
            .detectionAttributes(Attribute.DEFAULT)
            .build();

        IndexFacesResponse facesResponse = rekClient.indexFaces(facesRequest);
        System.out.println("Results for the image");
        System.out.println("\n Faces indexed:");
        List<FaceRecord> faceRecords = facesResponse.faceRecords();
        for (FaceRecord faceRecord : faceRecords) {
            System.out.println("  Face ID: " + faceRecord.face().faceId());
        }
    }
}
```

```
        System.out.println(" Location:" +
faceRecord.faceDetail().boundingBox().toString());
    }

    List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
    System.out.println("Faces not indexed:");
    for (UnindexedFace unindexedFace : unindexedFaces) {
        System.out.println(" Location:" +
unindexedFace.faceDetail().boundingBox().toString());
        System.out.println(" Reasons:");
        for (Reason reason : unindexedFace.reasons()) {
            System.out.println("Reason: " + reason);
        }
    }

} catch (RekognitionException | FileNotFoundException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス [IndexFaces](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListCollections

次の例は、ListCollections を使用する方法を説明しています。

コレクションの詳細については、「[コレクションを一覧表示する](#)」を参照してください。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

```
}
```

- API の詳細については、「API リファレンス [ListCollections](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListFaces

次の例は、ListFaces を使用する方法を説明しています。

詳細については、「[コレクションに顔を保存する](#)」を参照してください。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [ListFaces](#)」の「」を参照してください。AWS SDK for Java 2.x

## RecognizeCelebrities

次の例は、RecognizeCelebrities を使用する方法を説明しています。

詳細については、「[イメージ内で有名人を認識する](#)」を参照してください。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesRequest;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Celebrity;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RecognizeCelebrities {
    public static void main(String[] args) {
```

```
    final String usage = ""
        Usage:    <sourceImage>

        Where:
            sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String sourceImage = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Locating celebrities in " + sourceImage);
    recognizeAllCelebrities(rekClient, sourceImage);
    rekClient.close();
}

public static void recognizeAllCelebrities(RekognitionClient rekClient, String
sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        RecognizeCelebritiesRequest request =
RecognizeCelebritiesRequest.builder()
            .image(souImage)
            .build();

        RecognizeCelebritiesResponse result =
rekClient.recognizeCelebrities(request);
        List<Celebrity> celebs = result.celebrityFaces();
        System.out.println(celebs.size() + " celebrity(s) were recognized.\n");
        for (Celebrity celebrity : celebs) {
            System.out.println("Celebrity recognized: " + celebrity.name());
        }
    }
}
```



```
        System.out.println("Celebrity ID: " + celebrity.id());

        System.out.println("Further information (if available):");
        for (String url : celebrity.urls()) {
            System.out.println(url);
        }
        System.out.println();
    }
    System.out.println(result.unrecognizedFaces().size() + " face(s) were
unrecognized.");

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [RecognizeCelebrities](#)」の「」を参照してください。AWS SDK for Java 2.x

## SearchFaces

次の例は、SearchFaces を使用する方法を説明しています。

詳細については、[顔 \(フェイス ID\) を検索する](#) を参照してください。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
```

```
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingImageCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <sourceImage>

                Where:
                    collectionId - The id of the collection. \s
                    sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFaceInCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }
}
```

```
}

    public static void searchFaceInCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(new File(sourceImage));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        SearchFacesByImageRequest facesByImageRequest =
SearchFacesByImageRequest.builder()
            .image(souImage)
            .maxFaces(10)
            .faceMatchThreshold(70F)
            .collectionId(collectionId)
            .build();

        SearchFacesByImageResponse imageResponse =
rekClient.searchFacesByImage(facesByImageRequest);
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face : faceImageMatches) {
            System.out.println("The similarity level is " + face.similarity());
            System.out.println();
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [SearchFaces](#)」の「」を参照してください。AWS SDK for Java 2.x

## SearchFacesByImage

次の例は、SearchFacesByImage を使用する方法を説明しています。

詳細については、「[顔を検索する \(イメージ\)](#)」を参照してください。

## SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingIdCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <sourceImage>

                Where:
                    collectionId - The id of the collection. \s
                    sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String collectionId = args[0];
String faceId = args[1];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

System.out.println("Searching for a face in a collections");
searchFaceById(rekClient, collectionId, faceId);
rekClient.close();
}

public static void searchFaceById(RekognitionClient rekClient, String
collectionId, String faceId) {
    try {
        SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
            .collectionId(collectionId)
            .faceId(faceId)
            .faceMatchThreshold(70F)
            .maxFaces(2)
            .build();

        SearchFacesResponse imageResponse =
rekClient.searchFaces(searchFacesRequest);
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face : faceImageMatches) {
            System.out.println("The similarity level is " + face.similarity());
            System.out.println();
        }

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、「API リファレンス [SearchFacesByImage](#)」の「」を参照してください。AWS SDK for Java 2.x

## シナリオ

### ビデオ内の情報を検出する

次のコードサンプルは、以下の操作方法を示しています。

- Amazon Rekognition のジョブを開始し、人物、オブジェクト、テキストなどの要素を動画から検出します。
- ジョブが完了するまでジョブのステータスを確認します。
- 検出された要素のリストをジョブごとに出力します。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

Amazon S3 バケット内のビデオから有名人の結果を取得します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
import
    software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;
```

```
/**
 * To run this code example, ensure that you perform the Prerequisites as stated
 * in the Amazon Rekognition Guide:
 * https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-sqs.html
 *
 * Also, ensure that set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class VideoCelebrityDetection {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();
    }
}
```

```
NotificationChannel channel = NotificationChannel.builder()
    .snsTopicArn(topicArn)
    .roleArn(roleArn)
    .build();

startCelebrityDetection(rekClient, channel, bucket, video);
getCelebrityDetectionResults(rekClient);
System.out.println("This example is done!");
rekClient.close();
}

public static void startCelebrityDetection(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3obj)
            .build();

        StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
            .jobTag("Celebrities")
            .notificationChannel(channel)
            .video(vidObj)
            .build();

        StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient
            .startCelebrityRecognition(recognitionRequest);
        startJobId = startCelebrityRecognitionResult.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```



```
public static void getCelebrityDetectionResults(RekognitionClient rekClient) {

    try {
        String paginationToken = null;
        GetCelebrityRecognitionResponse recognitionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (recognitionResponse != null)
                paginationToken = recognitionResponse.nextToken();

            GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                .maxResults(10)
                .build();

            // Wait until the job succeeds
            while (!finished) {
                recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
                status = recognitionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }

            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null.
            VideoMetadata videoMetaData = recognitionResponse.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());
        }
    }
}
```

```
        System.out.println("Job");

        List<CelebrityRecognition> celebs =
recognitionResponse.celebrities();
        for (CelebrityRecognition celeb : celebs) {
            long seconds = celeb.timestamp() / 1000;
            System.out.print("Sec: " + seconds + " ");
            CelebrityDetail details = celeb.celebrity();
            System.out.println("Name: " + details.name());
            System.out.println("Id: " + details.id());
            System.out.println();
        }

        } while (recognitionResponse.nextToken() != null);

    } catch (RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

ラベル検出オペレーションによって、ビデオ内のラベルを検出します。

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
```

```
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of the video (for example, people.mp4).\s
                queueUrl- The URL of a SQS queue.\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
```

```
String topicArn = args[3];
String roleArn = args[4];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

SqsClient sqs = SqsClient.builder()
    .region(Region.US_EAST_1)
    .build();

NotificationChannel channel = NotificationChannel.builder()
    .snsTopicArn(topicArn)
    .roleArn(roleArn)
    .build();

startLabels(rekClient, channel, bucket, video);
getLabelJob(rekClient, sqs, queueUrl);
System.out.println("This example is done!");
sqs.close();
rekClient.close();
}

public static void startLabels(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vid0b = Video.builder()
            .s3object(s3obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vid0b)
            .minConfidence(50F)
            .build();
```

```
        StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

        boolean ans = true;
        String status = "";
        int yy = 0;
        while (ans) {

            GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .maxResults(10)
                .build();

            GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
            status = result.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                ans = false;
            else
                System.out.println(yy + " status is: " + status);

            Thread.sleep(1000);
            yy++;
        }

        System.out.println(startJobId + " status is: " + status);

    } catch (RekognitionException | InterruptedException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();
```

```
try {
    messages = sqs.receiveMessage(messageRequest).messages();

    if (!messages.isEmpty()) {
        for (Message message : messages) {
            String notification = message.body();

            // Get the status and job id from the notification
            ObjectMapper mapper = new ObjectMapper();
            JsonNode jsonMessageTree = mapper.readTree(notification);
            JsonNode messageBodyText = jsonMessageTree.get("Message");
            ObjectMapper operationResultMapper = new ObjectMapper();
            JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
            JsonNode operationJobId = jsonResultTree.get("JobId");
            JsonNode operationStatus = jsonResultTree.get("Status");
            System.out.println("Job found in JSON is " + operationJobId);

            DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

            String jobId = operationJobId.textValue();
            if (startJobId.compareTo(jobId) == 0) {
                System.out.println("Job id: " + operationJobId);
                System.out.println("Status : " +
operationStatus.toString());

                if (operationStatus.asText().equals("SUCCEEDED"))
                    getResultsLabels(rekClient);
                else
                    System.out.println("Video analysis failed");

                sqs.deleteMessage(deleteMessageRequest);
            } else {
                System.out.println("Job received was not job " +
startJobId);
                sqs.deleteMessage(deleteMessageRequest);
            }
        }
    }
} catch (RekognitionException e) {
```

```
        e.getMessage();
        System.exit(1);
    } catch (JsonMappingException e) {
        e.printStackTrace();
    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }
}

// Gets the job results by calling GetLabelDetection
private static void getResultsLabels(RecognitionClient rekClient) {

    int maxResults = 10;
    String paginationToken = null;
    GetLabelDetectionResponse labelDetectionResult = null;

    try {
        do {
            if (labelDetectionResult != null)
                paginationToken = labelDetectionResult.nextToken();

            GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .sortBy(LabelDetectionSortBy.TIMESTAMP)
                .maxResults(maxResults)
                .nextToken(paginationToken)
                .build();

            labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
            VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());

            List<LabelDetection> detectedLabels = labelDetectionResult.labels();
            for (LabelDetection detectedLabel : detectedLabels) {
                long seconds = detectedLabel.timestamp();
                Label label = detectedLabel.label();
                System.out.println("Millisecond: " + seconds + " ");

                System.out.println("  Label:" + label.name());
            }
        } while (labelDetectionResult.nextToken() != null);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
        System.out.println("    Confidence:" +
detectedLabel.label().confidence().toString());

        List<Instance> instances = label.instances();
        System.out.println("    Instances of " + label.name());

        if (instances.isEmpty()) {
            System.out.println("        " + "None");
        } else {
            for (Instance instance : instances) {
                System.out.println("            Confidence: " +
instance.confidence().toString());
                System.out.println("            Bounding box: " +
instance.boundingBox().toString());
            }
        }
        System.out.println("    Parent labels for " + label.name() +
");");

        List<Parent> parents = label.parents();

        if (parents.isEmpty()) {
            System.out.println("        None");
        } else {
            for (Parent parent : parents) {
                System.out.println("            " + parent.name());
            }
        }
        System.out.println();
    }
    } while (labelDetectionResult != null &&
labelDetectionResult.nextToken() != null);

    } catch (RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}
}
```

## Amazon S3 バケットに保存されたビデオ内の顔を検出する

```
import com.fasterxml.jackson.core.JsonProcessingException;
```



```
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

            Where:
```

```
        bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
        video - The name of the video (for example, people.mp4).\s
        queueUrl- The URL of a SQS queue.\s
        topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
        roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
        """;

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String queueUrl = args[2];
    String topicArn = args[3];
    String roleArn = args[4];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    SqsClient sqs = SqsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startLabels(rekClient, channel, bucket, video);
    getLabelJob(rekClient, sqs, queueUrl);
    System.out.println("This example is done!");
    sqs.close();
    rekClient.close();
}

public static void startLabels(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
```

```
String video) {
try {
    S3Object s3obj = S3Object.builder()
        .bucket(bucket)
        .name(video)
        .build();

    Video vid0b = Video.builder()
        .s3object(s3obj)
        .build();

    StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
    .jobTag("DetectingLabels")
    .notificationChannel(channel)
    .video(vid0b)
    .minConfidence(50F)
    .build();

    StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
    startJobId = labelDetectionResponse.jobId();

    boolean ans = true;
    String status = "";
    int yy = 0;
    while (ans) {

        GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
        .jobId(startJobId)
        .maxResults(10)
        .build();

        GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
        status = result.jobStatusAsString();

        if (status.compareTo("SUCCEEDED") == 0)
            ans = false;
        else
            System.out.println(yy + " status is: " + status);

        Thread.sleep(1000);
    }
}
```

```
        yy++;
    }

    System.out.println(startJobId + " status is: " + status);

} catch (RekognitionException | InterruptedException e) {
    e.getMessage();
    System.exit(1);
}
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();

        if (!messages.isEmpty()) {
            for (Message message : messages) {
                String notification = message.body();

                // Get the status and job id from the notification
                ObjectMapper mapper = new ObjectMapper();
                JsonNode jsonMessageTree = mapper.readTree(notification);
                JsonNode messageBodyText = jsonMessageTree.get("Message");
                ObjectMapper operationResultMapper = new ObjectMapper();
                JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                JsonNode operationJobId = jsonResultTree.get("JobId");
                JsonNode operationStatus = jsonResultTree.get("Status");
                System.out.println("Job found in JSON is " + operationJobId);

                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

                String jobId = operationJobId.textValue();
                if (startJobId.compareTo(jobId) == 0) {
                    System.out.println("Job id: " + operationJobId);
                }
            }
        }
    }
}
```

```
        System.out.println("Status : " +
operationStatus.toString());

        if (operationStatus.asText().equals("SUCCEEDED"))
            getResultsLabels(rekClient);
        else
            System.out.println("Video analysis failed");

        sqs.deleteMessage(deleteMessageRequest);
    } else {
        System.out.println("Job received was not job " +
startJobId);
        sqs.deleteMessage(deleteMessageRequest);
    }
}

} catch (RekognitionException e) {
    e.getMessage();
    System.exit(1);
} catch (JsonMappingException e) {
    e.printStackTrace();
} catch (JsonProcessingException e) {
    e.printStackTrace();
}
}

// Gets the job results by calling GetLabelDetection
private static void getResultsLabels(RekognitionClient rekClient) {

    int maxResults = 10;
    String paginationToken = null;
    GetLabelDetectionResponse labelDetectionResult = null;

    try {
        do {
            if (labelDetectionResult != null)
                paginationToken = labelDetectionResult.nextToken();

            GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .sortBy(LabelDetectionSortBy.TIMESTAMP)
                .maxResults(maxResults)
```

```
        .nextToken(paginationToken)
        .build();

    LabelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
    VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " + videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());

    List<LabelDetection> detectedLabels = labelDetectionResult.labels();
    for (LabelDetection detectedLabel : detectedLabels) {
        long seconds = detectedLabel.timestamp();
        Label label = detectedLabel.label();
        System.out.println("Millisecond: " + seconds + " ");

        System.out.println("    Label:" + label.name());
        System.out.println("    Confidence:" +
detectedLabel.label().confidence().toString());

        List<Instance> instances = label.instances();
        System.out.println("    Instances of " + label.name());

        if (instances.isEmpty()) {
            System.out.println("        " + "None");
        } else {
            for (Instance instance : instances) {
                System.out.println("        Confidence: " +
instance.confidence().toString());
                System.out.println("        Bounding box: " +
instance.boundingBox().toString());
            }
        }
        System.out.println("    Parent labels for " + label.name() +
":");

        List<Parent> parents = label.parents();

        if (parents.isEmpty()) {
            System.out.println("        None");
        } else {
            for (Parent parent : parents) {
                System.out.println("        " + parent.name());
            }
        }
    }
}
```

```
        }
        System.out.println();
    }
    } while (labelDetectionResult != null &&
labelDetectionResult.nextToken() != null);

    } catch (RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}
}
```

Amazon S3 バケットに保存されたビデオ内の不適切なコンテンツや攻撃的なコンテンツを検出します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;
import
    software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class VideoDetectInappropriate {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        NotificationChannel channel = NotificationChannel.builder()
            .snsTopicArn(topicArn)
            .roleArn(roleArn)
            .build();

        startModerationDetection(rekClient, channel, bucket, video);
        getModResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }
}
```



```
public static void startModerationDetection(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {

    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3obj)
            .build();

        StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
            .jobTag("Moderation")
            .notificationChannel(channel)
            .video(vidObj)
            .build();

        StartContentModerationResponse startModDetectionResult = rekClient
            .startContentModeration(modDetectionRequest);
        startJobId = startModDetectionResult.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void getModResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetContentModerationResponse modDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (modDetectionResponse != null)
                paginationToken = modDetectionResponse.nextToken();
        }
    }
}
```

```
        GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
    .jobId(startJobId)
    .nextToken(paginationToken)
    .maxResults(10)
    .build();

    // Wait until the job succeeds.
    while (!finished) {
        modDetectionResponse =
rekClient.getContentModeration(modRequest);
        status = modDetectionResponse.jobStatusAsString();

        if (status.compareTo("SUCCEEDED") == 0)
            finished = true;
        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null.
    VideoMetadata videoMetaData = modDetectionResponse.videoMetadata();
    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " + videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());
    System.out.println("Job");

    List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
    for (ContentModerationDetection mod : mods) {
        long seconds = mod.timestamp() / 1000;
        System.out.print("Mod label: " + seconds + " ");
        System.out.println(mod.moderationLabel().toString());
        System.out.println();
    }

    } while (modDetectionResponse != null &&
modDetectionResponse.nextToken() != null);
```

```
        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Amazon S3 バケットに保存されているビデオ内のテクニカルキューセグメントおよびショット検出セグメントを検出します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartShotDetectionFilter;
import
    software.amazon.awssdk.services.rekognition.model.StartTechnicalCueDetectionFilter;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionFilters;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.SegmentDetection;
import software.amazon.awssdk.services.rekognition.model.TechnicalCueSegment;
import software.amazon.awssdk.services.rekognition.model.ShotSegment;
import software.amazon.awssdk.services.rekognition.model.SegmentType;
import software.amazon.awssdk.services.sqs.SqsClient;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class VideoDetectSegment {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        SqsClient sqs = SqsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        NotificationChannel channel = NotificationChannel.builder()
            .snsTopicArn(topicArn)
            .roleArn(roleArn)
            .build();
    }
}
```

```
        startSegmentDetection(rekClient, channel, bucket, video);
        getSegmentResults(rekClient);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startSegmentDetection(RecognitionClient rekClient,
        NotificationChannel channel,
        String bucket,
        String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                .bucket(bucket)
                .name(video)
                .build();

            Video vidObj = Video.builder()
                .s3Object(s3Obj)
                .build();

            StartShotDetectionFilter cueDetectionFilter =
            StartShotDetectionFilter.builder()
                .minSegmentConfidence(60F)
                .build();

            StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
            StartTechnicalCueDetectionFilter.builder()
                .minSegmentConfidence(60F)
                .build();

            StartSegmentDetectionFilters filters =
            StartSegmentDetectionFilters.builder()
                .shotFilter(cueDetectionFilter)
                .technicalCueFilter(technicalCueDetectionFilter)
                .build();

            StartSegmentDetectionRequest segDetectionRequest =
            StartSegmentDetectionRequest.builder()
                .jobTag("DetectingLabels")
                .notificationChannel(channel)
                .segmentTypes(SegmentType.TECHNICAL_CUE, SegmentType.SHOT)
                .video(vidObj)
                .filters(filters)
        }
```

```
        .build();

        StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
        startJobId = segDetectionResponse.jobId();

    } catch (RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getSegmentResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetSegmentDetectionResponse segDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (segDetectionResponse != null)
                paginationToken = segDetectionResponse.nextToken();

            GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {
                segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
                status = segDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }
        }
    }
}
```

```
    }
    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null.
    List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
    for (VideoMetadata metaData : videoMetaData) {
        System.out.println("Format: " + metaData.format());
        System.out.println("Codec: " + metaData.codec());
        System.out.println("Duration: " + metaData.durationMillis());
        System.out.println("FrameRate: " + metaData.frameRate());
        System.out.println("Job");
    }

    List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
    for (SegmentDetection detectedSegment : detectedSegments) {
        String type = detectedSegment.type().toString();
        if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
            System.out.println("Technical Cue");
            TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
            System.out.println("\tType: " + segmentCue.type());
            System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
        }

        if (type.contains(SegmentType.SHOT.toString())) {
            System.out.println("Shot");
            ShotSegment segmentShot = detectedSegment.shotSegment();
            System.out.println("\tIndex " + segmentShot.index());
            System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
        }

        long seconds = detectedSegment.durationMillis();
        System.out.println("\tDuration : " + seconds + " milliseconds");
        System.out.println("\tStart time code: " +
detectedSegment.startTimecodeSMPTE());
        System.out.println("\tEnd time code: " +
detectedSegment.endTimecodeSMPTE());
        System.out.println("\tDuration time code: " +
detectedSegment.durationSMPTE());
        System.out.println();
    }
}
```

```
        }

        } while (segDetectionResponse != null &&
segDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Amazon S3 バケットに保存されたビデオに保存されたビデオ内のテキストを検出します。

```
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 vid0b = Video.builder()
        .s3object(s3obj)
        .build();

    StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
        .jobTag("DetectingLabels")
        .notificationChannel(channel)
        .video(vid0b)
        .build();

    StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
    startJobId = labelDetectionResponse.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void getTextResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetTextDetectionResponse textDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (textDetectionResponse != null)
                paginationToken = textDetectionResponse.nextToken();

            GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();
```

```
        // Wait until the job succeeds.
        while (!finished) {
            textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
            status = textDetectionResponse.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(yy + " status is: " + status);
                Thread.sleep(1000);
            }
            yy++;
        }

        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is null.
        VideoMetadata videoMetaData = textDetectionResponse.videoMetadata();
        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " + videoMetaData.durationMillis());
        System.out.println("FrameRate: " + videoMetaData.frameRate());
        System.out.println("Job");

        List<TextDetectionResult> labels =
textDetectionResponse.textDetections();
        for (TextDetectionResult detectedText : labels) {
            System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
            System.out.println("Id : " + detectedText.textDetection().id());
            System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
            System.out.println("Type: " +
detectedText.textDetection().type());
            System.out.println("Text: " +
detectedText.textDetection().detectedText());
            System.out.println();
        }

        } while (textDetectionResponse != null &&
textDetectionResponse.nextToken() != null);

    } catch (RekognitionException | InterruptedException e) {
```

```
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

Amazon S3 バケットに保存されたビデオに保存されたビデオ内の人物を検出します。

```
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 vid0b = Video.builder()
            .s3object(s3obj)
            .build();

        StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
            .jobTag("DetectingLabels")
            .video(vid0b)
            .notificationChannel(channel)
            .build();

        StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
        startJobId = labelDetectionResponse.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void getPersonDetectionResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetPersonTrackingResponse personTrackingResult = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (personTrackingResult != null)
                paginationToken = personTrackingResult.nextToken();

            GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds
            while (!finished) {
```

```
        personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
        status = personTrackingResult.jobStatusAsString();

        if (status.compareTo("SUCCEEDED") == 0)
            finished = true;
        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null.
    VideoMetadata videoMetaData = personTrackingResult.videoMetadata();

    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " + videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());
    System.out.println("Job");

    List<PersonDetection> detectedPersons =
personTrackingResult.persons();
    for (PersonDetection detectedPerson : detectedPersons) {
        long seconds = detectedPerson.timestamp() / 1000;
        System.out.print("Sec: " + seconds + " ");
        System.out.println("Person Identifier: " +
detectedPerson.person().index());
        System.out.println();
    }

    } while (personTrackingResult != null &&
personTrackingResult.nextToken() != null);

    } catch (RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [GetCelebrityRecognition](#)
  - [GetContentModeration](#)
  - [GetLabelDetection](#)
  - [GetPersonTracking](#)
  - [GetSegmentDetection](#)
  - [GetTextDetection](#)
  - [StartCelebrityRecognition](#)
  - [StartContentModeration](#)
  - [StartLabelDetection](#)
  - [StartPersonTracking](#)
  - [StartSegmentDetection](#)
  - [StartTextDetection](#)

## SDK for Java 2.x を使用した Route 53 ドメイン登録のコード例

次のコード例は、Route 53 ドメイン登録 AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

### 開始方法

#### ハロー Route 53 ドメイン登録

以下のコード例は、Route 53 ドメイン登録の使用を開始する方法を示しています。



## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.route53domains.Route53DomainsClient;
import software.amazon.awssdk.services.route53.model.Route53Exception;
import software.amazon.awssdk.services.route53domains.model.DomainPrice;
import software.amazon.awssdk.services.route53domains.model.ListPricesRequest;
import software.amazon.awssdk.services.route53domains.model.ListPricesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code examples performs the following operation:
 *
 * 1. Invokes ListPrices for at least one domain type, such as the "com" type
 * and displays the prices for Registration and Renewal.
 */
public class HelloRoute53 {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <hostedZoneId> \n\n" +
            "Where:\n" +
            "    hostedZoneId - The id value of an existing hosted zone. \n";

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String domainType = args[0];
    Region region = Region.US_EAST_1;
    Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Invokes ListPrices for at least one domain type.");
    listPrices(route53DomainsClient, domainType);
    System.out.println(DASHES);
}

public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
    try {
        ListPricesRequest pricesRequest = ListPricesRequest.builder()
            .maxItems(10)
            .tld(domainType)
            .build();

        ListPricesResponse response =
route53DomainsClient.listPrices(pricesRequest);
        List<DomainPrice> prices = response.prices();
        for (DomainPrice pr : prices) {
            System.out.println("Name: " + pr.name());
            System.out.println(
                "Registration: " + pr.registrationPrice().price() + " " +
pr.registrationPrice().currency());
            System.out.println("Renewal: " + pr.renewalPrice().price() + " " +
pr.renewalPrice().currency());
            System.out.println("Transfer: " + pr.transferPrice().price() + " " +
pr.transferPrice().currency());
            System.out.println("Transfer: " + pr.transferPrice().price() + " " +
pr.transferPrice().currency());
            System.out.println("Change Ownership: " +
pr.changeOwnershipPrice().price() + " "
                + pr.changeOwnershipPrice().currency());
            System.out.println(
                "Restoration: " + pr.restorationPrice().price() + " " +
pr.restorationPrice().currency());
            System.out.println(" ");
        }
    }
}
```

```
    }  
  
    } catch (Route53Exception e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}  
}
```

- API の詳細については、「API リファレンス [ListPrices](#)」の「」を参照してください。AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### CheckDomainAvailability

次の例は、CheckDomainAvailability を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void checkDomainAvailability(Route53DomainsClient  
route53DomainsClient, String domainSuggestion) {  
    try {  
        CheckDomainAvailabilityRequest availabilityRequest =  
CheckDomainAvailabilityRequest.builder()  
            .domainName(domainSuggestion)  
            .build();  
  
        CheckDomainAvailabilityResponse response = route53DomainsClient
```

```
        .checkDomainAvailability(availabilityRequest);
        System.out.println(domainSuggestion + " is " +
response.availability().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス[CheckDomainAvailability](#)」の「」を参照してください。AWS SDK for Java 2.x

## CheckDomainTransferability

次の例は、CheckDomainTransferability を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainTransferabilityResponse response = route53DomainsClient
            .checkDomainTransferability(transferabilityRequest);
        System.out.println("Transferability: " +
response.transferability().transferable().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [CheckDomainTransferability](#)」の「」を参照してください。AWS SDK for Java 2.x

## GetDomainDetail

次の例は、GetDomainDetail を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
    try {
        GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
            .domainName(domainSuggestion)
            .build();

        GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
        System.out.println("The contact first name is " +
response.registrantContact().firstName());
        System.out.println("The contact last name is " +
response.registrantContact().lastName());
        System.out.println("The contact org name is " +
response.registrantContact().organizationName());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [GetDomainDetail](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## GetDomainSuggestions

次の例は、GetDomainSuggestions を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        GetDomainSuggestionsRequest suggestionsRequest =
        GetDomainSuggestionsRequest.builder()
            .domainName(domainSuggestion)
            .suggestionCount(5)
            .onlyAvailable(true)
            .build();

        GetDomainSuggestionsResponse response =
        route53DomainsClient.getDomainSuggestions(suggestionsRequest);
        List<DomainSuggestion> suggestions = response.suggestionsList();
        for (DomainSuggestion suggestion : suggestions) {
            System.out.println("Suggestion Name: " + suggestion.domainName());
            System.out.println("Availability: " + suggestion.availability());
            System.out.println(" ");
        }

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [GetDomainSuggestions](#)」の「」を参照してください。AWS SDK for Java 2.x

## GetOperationDetail

次の例は、GetOperationDetail を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
    try {
        GetOperationDetailRequest detailRequest =
        GetOperationDetailRequest.builder()
            .operationId(operationId)
            .build();

        GetOperationDetailResponse response =
        route53DomainsClient.getOperationDetail(detailRequest);
        System.out.println("Operation detail message is " + response.message());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [GetOperationDetail](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListDomains

次の例は、ListDomains を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void listDomains(Route53DomainsClient route53DomainsClient) {
    try {
        ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
        listRes.stream()
            .flatMap(r -> r.domains().stream())
            .forEach(content -> System.out.println("The domain name is " +
content.domainName()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [ListDomains](#)」の「」を参照してください。 AWS SDK for Java 2.x

## ListOperations

次の例は、ListOperations を使用する方法を説明しています。



## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void listOperations(Route53DomainsClient route53DomainsClient) {
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        localDateTime = localDateTime.minusYears(1);
        Instant myTime = localDateTime.toInstant(zoneOffset);

        ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
            .submittedSince(myTime)
            .build();

        ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
        listRes.stream()
            .flatMap(r -> r.operations().stream())
            .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                " Status: " + content.statusAsString() +
                " Date: " + content.submittedDate()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [ListOperations](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListPrices

次の例は、ListPrices を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
    try {
        ListPricesRequest pricesRequest = ListPricesRequest.builder()
            .tld(domainType)
            .build();

        ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
        listRes.stream()
            .flatMap(r -> r.prices().stream())
            .forEach(content -> System.out.println(" Name: " +
content.name() +
                " Registration: " + content.registrationPrice().price()
+ " "
                + content.registrationPrice().currency() +
                " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [ListPrices](#)」の「」を参照してください。 AWS SDK for Java 2.x

## RegisterDomain

次の例は、RegisterDomain を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
    String domainSuggestion,
    String phoneNumber,
    String email,
    String firstName,
    String lastName,
    String city) {

    try {
        ContactDetail contactDetail = ContactDetail.builder()
            .contactType(ContactType.COMPANY)
            .state("LA")
            .countryCode(CountryCode.IN)
            .email(email)
            .firstName(firstName)
            .lastName(lastName)
            .city(city)
            .phoneNumber(phoneNumber)
            .organizationName("My Org")
            .addressLine1("My Address")
            .zipCode("123 123")
            .build();

        RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
            .adminContact(contactDetail)
            .registrantContact(contactDetail)
            .techContact(contactDetail)
            .domainName(domainSuggestion)
            .autoRenew(true)
            .durationInYears(1)
```

```
        .build();

        RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
        System.out.println("Registration requested. Operation Id: " +
response.operationId());
        return response.operationId();

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- API の詳細については、「API リファレンス [RegisterDomain](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ViewBilling

次の例は、ViewBilling を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        LocalDateTime localDateTime2 = localDateTime.minusYears(1);
        Instant myStartTime = localDateTime2.toInstant(zoneOffset);
        Instant myEndTime = localDateTime.toInstant(zoneOffset);
```

```
ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
    .start(myStartTime)
    .end(myEndTime)
    .build();

ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
listRes.stream()
    .flatMap(r -> r.billingRecords().stream())
    .forEach(content -> System.out.println(" Bill Date: " +
content.billDate() +
        " Operation: " + content.operationAsString() +
        " Price: " + content.price()));

} catch (Route53Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス [ViewBilling](#)」の「」を参照してください。AWS SDK for Java 2.x

## シナリオ

### ドメインを始める

次のコードサンプルは、以下の操作方法を示しています。

- 現在のドメインを一覧表示し、過去 1 年間の操作を一覧表示します。
- 過去 1 年間の請求記録とドメインタイプの価格を表示します。
- ドメインの候補を取得します。
- ドメインの可用性と移管可能性を確認します。
- オプションで、ドメイン登録をリクエストします。
- 操作の詳細を入手します。
- オプションで、ドメインの詳細を取得します。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example uses pagination methods where applicable. For example, to list
 * domains, the
 * listDomainsPaginator method is used. For more information about pagination,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/pagination.html
 *
 * This Java code example performs the following operations:
 *
 * 1. List current domains.
 * 2. List operations in the past year.
 * 3. View billing for the account in the past year.
 * 4. View prices for domain types.
 * 5. Get domain suggestions.
 * 6. Check domain availability.
 * 7. Check domain transferability.
 * 8. Request a domain registration.
 * 9. Get operation details.
 * 10. Optionally, get domain details.
 */

public class Route53Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = ""
```

```
Usage:
    <domainType> <phoneNumber> <email> <domainSuggestion>
<firstName> <lastName> <city>

Where:
    domainType - The domain type (for example, com).\s
    phoneNumber - The phone number to use (for example,
+91.9966564xxx)    email - The email address to use.    domainSuggestion - The
domain suggestion (for example, findmy.accountants).\s
    firstName - The first name to use to register a domain.\s
    lastName - The last name to use to register a domain.\s
    city - the city to use to register a domain.\s
    """;

if (args.length != 7) {
    System.out.println(usage);
    System.exit(1);
}

String domainType = args[0];
String phoneNumber = args[1];
String email = args[2];
String domainSuggestion = args[3];
String firstName = args[4];
String lastName = args[5];
String city = args[6];
Region region = Region.US_EAST_1;
Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon Route 53 domains example
scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. List current domains.");
listDomains(route53DomainsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. List operations in the past year.");
```

```
listOperations(route53DomainsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. View billing for the account in the past year.");
listBillingRecords(route53DomainsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. View prices for domain types.");
listPrices(route53DomainsClient, domainType);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get domain suggestions.");
listDomainSuggestions(route53DomainsClient, domainSuggestion);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Check domain availability.");
checkDomainAvailability(route53DomainsClient, domainSuggestion);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Check domain transferability.");
checkDomainTransferability(route53DomainsClient, domainSuggestion);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Request a domain registration.");
String opId = requestDomainRegistration(route53DomainsClient,
    domainSuggestion, phoneNumber, email, firstName,
    lastName, city);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Get operation details.");
getOperationalDetail(route53DomainsClient, opId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get domain details.");
System.out.println("Note: You must have a registered domain to get
details.");
```



```
        System.out.println("Otherwise, an exception is thrown that states ");
        System.out.println("Domain xxxxxxxx not found in xxxxxxxx account.");
        getDomainDetails(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);
    }

    public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
        try {
            GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
                .domainName(domainSuggestion)
                .build();

            GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
            System.out.println("The contact first name is " +
response.registrantContact().firstName());
            System.out.println("The contact last name is " +
response.registrantContact().lastName());
            System.out.println("The contact org name is " +
response.registrantContact().organizationName());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
        try {
            GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
                .operationId(operationId)
                .build();

            GetOperationDetailResponse response =
route53DomainsClient.getOperationDetail(detailRequest);
            System.out.println("Operation detail message is " + response.message());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

```
}

    public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
        String domainSuggestion,
        String phoneNumber,
        String email,
        String firstName,
        String lastName,
        String city) {

    try {
        ContactDetail contactDetail = ContactDetail.builder()
            .contactType(ContactType.COMPANY)
            .state("LA")
            .countryCode(CountryCode.IN)
            .email(email)
            .firstName(firstName)
            .lastName(lastName)
            .city(city)
            .phoneNumber(phoneNumber)
            .organizationName("My Org")
            .addressLine1("My Address")
            .zipCode("123 123")
            .build();

        RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
            .adminContact(contactDetail)
            .registrantContact(contactDetail)
            .techContact(contactDetail)
            .domainName(domainSuggestion)
            .autoRenew(true)
            .durationInYears(1)
            .build();

        RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
        System.out.println("Registration requested. Operation Id: " +
response.operationId());
        return response.operationId();

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }
    return "";
}

public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainTransferabilityResponse response = route53DomainsClient
            .checkDomainTransferability(transferabilityRequest);
        System.out.println("Transferability: " +
response.transferability().transferable().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainAvailabilityResponse response = route53DomainsClient
            .checkDomainAvailability(availabilityRequest);
        System.out.println(domainSuggestion + " is " +
response.availability().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
```

```
        try {
            GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
                .domainName(domainSuggestion)
                .suggestionCount(5)
                .onlyAvailable(true)
                .build();

            GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
            List<DomainSuggestion> suggestions = response.suggestionsList();
            for (DomainSuggestion suggestion : suggestions) {
                System.out.println("Suggestion Name: " + suggestion.domainName());
                System.out.println("Availability: " + suggestion.availability());
                System.out.println(" ");
            }

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                .tld(domainType)
                .build();

            ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
            listRes.stream()
                .flatMap(r -> r.prices().stream())
                .forEach(content -> System.out.println(" Name: " +
content.name() +
                    " Registration: " + content.registrationPrice().price()
+ " "
                    + content.registrationPrice().currency() +
                    " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
        }
    }
}
```

```
        System.exit(1);
    }
}

public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        LocalDateTime localDateTime2 = localDateTime.minusYears(1);
        Instant myStartTime = localDateTime2.toInstant(zoneOffset);
        Instant myEndTime = localDateTime.toInstant(zoneOffset);

        ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
            .start(myStartTime)
            .end(myEndTime)
            .build();

        ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
        listRes.stream()
            .flatMap(r -> r.billingRecords().stream())
            .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                " Operation: " + content.operationAsString() +
                " Price: " + content.price()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listOperations(Route53DomainsClient route53DomainsClient) {
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        localDateTime = localDateTime.minusYears(1);
        Instant myTime = localDateTime.toInstant(zoneOffset);
```

```
        ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
        .submittedSince(myTime)
        .build();

        ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
        listRes.stream()
        .flatMap(r -> r.operations().stream())
        .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                " Status: " + content.statusAsString() +
                " Date: " + content.submittedDate()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listDomains(Route53DomainsClient route53DomainsClient) {
    try {
        ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
        listRes.stream()
        .flatMap(r -> r.domains().stream())
        .forEach(content -> System.out.println("The domain name is " +
content.domainName()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [CheckDomainAvailability](#)
  - [CheckDomainTransferability](#)
  - [GetDomainDetail](#)

- [GetDomainSuggestions](#)
- [GetOperationDetail](#)
- [ListDomains](#)
- [ListOperations](#)
- [ListPrices](#)
- [RegisterDomain](#)
- [ViewBilling](#)

## SDK for Java 2.x を使用する Amazon S3 の例

次のコード例は、Amazon S3 AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

### 開始方法

Hello Amazon S3

次のコード例は、Amazon S3 の使用を開始する方法を示しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }

    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- APIの詳細については、「APIリファレンス[ListBuckets](#)」の「」を参照してください。AWS SDK for Java 2.x



## トピック

- [アクション](#)
- [シナリオ](#)
- [サーバーレスサンプル](#)

## アクション

### CopyObject

次の例は、CopyObject を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

[S3Client](#) を使用してオブジェクトをコピーします。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CopyObject {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
```

```
        <objectKey> <fromBucket> <toBucket>

    Where:
        objectKey - The name of the object (for example, book.pdf).
        fromBucket - The S3 bucket name that contains the object (for
example, bucket1).
        toBucket - The S3 bucket to copy the object to (for example,
bucket2).

        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String objectKey = args[0];
    String fromBucket = args[1];
    String toBucket = args[2];
    System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    copyBucketObject(s3, fromBucket, objectKey, toBucket);
    s3.close();
}

public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .sourceBucket(fromBucket)
        .sourceKey(objectKey)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        return copyRes.copyObjectResult().toString();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
    return "";
}
}
```

[S3TransferManager](#) を使用して、あるバケットから別のバケットに[オブジェクトをコピー](#)します。[完全なファイル](#)と[テスト](#)を表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

public String copyObject(S3TransferManager transferManager, String bucketName,
    String key, String destinationBucket, String destinationKey) {
    CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
        .sourceBucket(bucketName)
        .sourceKey(key)
        .destinationBucket(destinationBucket)
        .destinationKey(destinationKey)
        .build();

    CopyRequest copyRequest = CopyRequest.builder()
        .copyObjectRequest(copyObjectRequest)
        .build();

    Copy copy = transferManager.copy(copyRequest);

    CompletedCopy completedCopy = copy.completionFuture().join();
    return completedCopy.response().copyObjectResult().eTag();
}
```

- API の詳細については、「API リファレンス[CopyObject](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateBucket

次の例は、CreateBucket を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

バケットを作成します。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateBucket {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

                Usage:
                <bucketName>\s

                Where:
                bucketName - The name of the bucket to create. The bucket name
                must be unique, or an error occurs.
    }
}
```

```
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    System.out.format("Creating a bucket named %s\n", bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    createBucket(s3, bucketName);
    s3.close();
}

public static void createBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

オブジェクトロックを有効にしてバケットを作成します。

```
// Create a new Amazon S3 bucket with object lock options.
public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
    S3Waiter s3Waiter = getClient().waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .objectLockEnabledForBucket(enableObjectLock)
        .build();

    getClient().createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    System.out.println(bucketName + " is ready");
}
```

- API の詳細については、「API リファレンス [CreateBucket](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteBucket

次の例は、DeleteBucket を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();
```

```
s3.deleteBucket(deleteBucketRequest);
s3.close();
```

- API の詳細については、「API リファレンス [DeleteBucket](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteBucketPolicy

次の例は、DeleteBucketPolicy を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = ""

                Usage:
                <bucketName>
```

```
        Where:
            bucketName - The Amazon S3 bucket to delete the policy from (for
example, bucket1).""";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    deleteS3BucketPolicy(s3, bucketName);
    s3.close();
}

// Delete the bucket policy.
public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
    DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketPolicy(delReq);
        System.out.println("Done!");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、「APIリファレンス[DeleteBucketPolicy](#)」の「」を参照してください。AWS SDK for Java 2.x



## DeleteBucketWebsite

次の例は、DeleteBucketWebsite を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

                Usage:      <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the website
configuration from.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
```

```
        System.out.format("Deleting website configuration for Amazon S3 bucket: %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }

    public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName) {
        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            s3.deleteBucketWebsite(delReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.out.println("Failed to delete website configuration!");
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [DeleteBucketWebsite](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteObjects

次の例は、DeleteObjects を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <bucketName>

                Where:
                    bucketName - the Amazon S3 bucket name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String bucketName = args[0];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

deleteBucketObjects(s3, bucketName);
s3.close();
}

public static void deleteBucketObjects(S3Client s3, String bucketName) {
    // Upload three sample objects to the specified Amazon S3 bucket.
    ArrayList<ObjectIdentifier> keys = new ArrayList<>();
    PutObjectRequest putOb;
    ObjectIdentifier objectId;

    for (int i = 0; i < 3; i++) {
        String keyName = "delete object example " + i;
        objectId = ObjectIdentifier.builder()
            .key(keyName)
            .build();

        putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        s3.putObject(putOb, RequestBody.fromString(keyName));
        keys.add(objectId);
    }

    System.out.println(keys.size() + " objects successfully created.");

    // Delete multiple objects in one request.
    Delete del = Delete.builder()
        .objects(keys)
        .build();

    try {
        DeleteObjectsRequest multiObjectDeleteRequest =
DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(del)
            .build();
```

```
s3.deleteObjects(multiObjectDeleteRequest);
System.out.println("Multiple objects are deleted!");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [DeleteObjects](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## GetBucketAcl

次の例は、GetBucketAcl を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
```

```
public class GetAcl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <objectKey>

            Where:
                bucketName - The Amazon S3 bucket to get the access control list
(ACL) for.
                objectKey - The object to get the ACL for.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    public static String getBucketACL(S3Client s3, String objectKey, String
bucketName) {
        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
```

```
        List<Grant> grants = aclRes.grants();
        String grantee = "";
        for (Grant grant : grants) {
            System.out.format("  %s: %s\n", grant.grantee().id(),
grant.permission());
            grantee = grant.grantee().id();
        }

        return grantee;
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
}
```

- API の詳細については、「API リファレンス [GetBucketAcl](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## GetBucketPolicy

次の例は、GetBucketPolicy を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
```

```
public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to get the policy from.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }

    public static String getPolicy(S3Client s3, String bucketName) {
        String policyText;
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();

        try {
```



```
        GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
        policyText = policyRes.policy();
        return policyText;

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
}
```

- API の詳細については、「API リファレンス [GetBucketPolicy](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## GetObject

次の例は、GetObject を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

[S3Client](#) を使用して、データをバイト配列として読み取ります。

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getObjectBytes(s3, bucketName, keyName, path);
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)

```

```
        .bucket(bucketName)
        .build();

    ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
    byte[] data = objectBytes.asByteArray();

    // Write the data to a local file.
    File myFile = new File(path);
    OutputStream os = new FileOutputStream(myFile);
    os.write(data);
    System.out.println("Successfully obtained bytes from an S3 object");
    os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

[S3TransferManager](#) を使用して、S3 バケット内の [オブジェクトをローカルファイルにダウンロード](#) します。 [完全なファイル](#) と [テスト](#) を表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
```

```

import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String bucketName,
                            String key, String downloadedFilePath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
            .getObjectRequest(b -> b.bucket(bucketName).key(key))
            .destination(Paths.get(downloadedFilePath))
            .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
downloadResult.response().contentType());
        return downloadResult.response().contentType();
    }

```

[S3Client](#) を使用して、オブジェクトに属するタグを読み取ります。

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = ""

```

```
Usage:
    <bucketName> <keyName>\s

Where:
    bucketName - The Amazon S3 bucket name.\s
    keyName - A key name that represents the object.\s
""";

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String bucketName = args[0];
String keyName = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

listTags(s3, bucketName, keyName);
s3.close();
}

public static void listTags(S3Client s3, String bucketName, String keyName) {
    try {
        GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        GetObjectTaggingResponse tags = s3.getObjectTagging(getTaggingRequest);
        List<Tag> tagSet = tags.tagSet();
        for (Tag tag : tagSet) {
            System.out.println(tag.key());
            System.out.println(tag.value());
        }
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

[S3Client](#) を使用してオブジェクトの URL を取得します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectUrl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
```

```
        getURL(s3, bucketName, keyName);
        s3.close();
    }

    public static void getURL(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            URL url = s3.utilities().getUrl(request);
            System.out.println("The URL for " + keyName + " is " + url);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

[S3Client](#) を使用する S3Presigner クライアントオブジェクトを使用してオブジェクトを取得します。

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents a text file.\s
            """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
            .region(region)
            .build();

        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(60))
                .getObjectRequest(getObjectRequest)
```



```
        .build();

        PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
        String theUrl = presignedGetObjectRequest.url().toString();
        System.out.println("Presigned URL: " + theUrl);
        HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
        presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
            values.forEach(value -> {
                connection.setRequestProperty(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();
    }
}
}
```

オブジェクトと [S3Client](#) を使用して ResponseTransformer オブジェクトを取得します。

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetDataResponseTransformer {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
```

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

getObjectBytes(s3, bucketName, keyName, path);
s3.close();
}

public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3.getObject(objectRequest, ResponseTransformer.toBytes());
        byte[] data = objectBytes.asByteArray();

        // Write the data to a local file.
        File myFile = new File(path);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
        System.out.println("Successfully obtained bytes from an S3 object");
        os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、「APIリファレンス[GetObject](#)」の「」を参照してください。AWS SDK for Java 2.x

## GetObjectLegalHold

次の例は、GetObjectLegalHold を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
            ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();

    } catch (S3Exception ex) {
        System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
        "'");
    }

    return null;
}
```

- API の詳細については、「API リファレンス [GetObjectLegalHold](#)」の「」を参照してください。 AWS SDK for Java 2.x

## GetObjectLockConfiguration

次の例は、GetObjectLockConfiguration を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Get the object lock configuration details for an S3 bucket.
public void getBucketObjectLockConfiguration(String bucketName) {
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .build();

    GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
    System.out.println("Bucket object lock config for "+bucketName+": ");
    System.out.println("\tEnabled:
"+response.getObjectLockConfiguration().getObjectLockEnabled());
    System.out.println("\tRule: "+
response.getObjectLockConfiguration().rule().defaultRetention());
}
```

- API の詳細については、「API リファレンス [GetObjectLockConfiguration](#)」の「」を参照してください。 AWS SDK for Java 2.x

## GetObjectRetention

次の例は、GetObjectRetention を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Get the retention period for an S3 object.
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {
        GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

        GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
        System.out.println("Object retention for "+key+" in "+ bucketName +":
"+ response.retention().mode() +" until "+ response.retention().retainUntilDate()
+ ".");

        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}
```

- API の詳細については、「API リファレンス [GetObjectRetention](#)」の「」を参照してください。 AWS SDK for Java 2.x

## HeadObject

次の例は、HeadObject を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## オブジェクトのコンテンツタイプを調べる

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <keyName>>

                Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
```

```
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }

    public static void getContentType(S3Client s3, String bucketName, String
keyName) {
        try {
            HeadObjectRequest objectRequest = HeadObjectRequest.builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            HeadObjectResponse objectHead = s3.headObject(objectRequest);
            String type = objectHead.contentType();
            System.out.println("The object content type is " + type);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

## オブジェクトの復元ステータスを取得する

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <keyName>\s
```



```
        Where:
            bucketName - The Amazon S3 bucket name.\s
            keyName - A key name that represents the object.\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String keyName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    checkStatus(s3, bucketName, keyName);
    s3.close();
}

public static void checkStatus(S3Client s3, String bucketName, String keyName) {
    try {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        HeadObjectResponse response = s3.headObject(headObjectRequest);
        System.out.println("The Amazon S3 object restoration status is " +
response.restore());

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [HeadObject](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListBuckets

次の例は、ListBuckets を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);
    }

    public static void listAllBuckets(S3Client s3) {
        ListBucketsResponse response = s3.listBuckets();
        List<Bucket> bucketList = response.buckets();
        for (Bucket bucket: bucketList) {
            System.out.println("Bucket name "+bucket.name());
        }
    }
}
```

```
}
```

- API の詳細については、「API リファレンス [ListBuckets](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListMultipartUploads

次の例は、ListMultipartUploads を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName>\s
```

```
        Where:
            bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();
    listUploads(s3, bucketName);
    s3.close();
}

public static void listUploads(S3Client s3, String bucketName) {
    try {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

        ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();
        for (MultipartUpload upload : uploads) {
            System.out.println("Upload in progress: Key = \"" + upload.key() +
"\", id = " + upload.uploadId());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [ListMultipartUploads](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListObjectsV2

次の例は、ListObjectsV2 を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListObjects {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName>\s

                Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
        """;
```

```
    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    listBucketObjects(s3, bucketName);
    s3.close();
}

public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.print("\n The name of the key is " + myValue.key());
            System.out.print("\n The object is " + calKb(myValue.size()) + "
KBs");

            System.out.print("\n The owner is " + myValue.owner());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// convert bytes to kbs.
private static long calKb(Long val) {
    return val / 1024;
}
}
```

ページネーションを使用してオブジェクトを一覧表示します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();
        }
    }
}
```

```
ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
listRes.stream()
    .flatMap(r -> r.contents().stream())
    .forEach(content -> System.out.println(" Key: " + content.key()
+ " size = " + content.size()));

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス」の [ListObjectsV2](#) を参照してください。AWS SDK for Java 2.x

## PutBucketAcl

次の例は、PutBucketAcl を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;
```



```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <id>\s

            Where:
                bucketName - The Amazon S3 bucket to grant permissions on.\s
                id - The ID of the owner of this bucket (you can get this value
from the AWS Management Console).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setBucketAcl(s3, bucketName, id);
        System.out.println("Done!");
        s3.close();
    }

    public static void setBucketAcl(S3Client s3, String bucketName, String id) {
        try {
            Grant ownerGrant = Grant.builder()
                .grantee(builder -> builder.id(id)

```

```
                .type(Type.CANONICAL_USER))
                .permission(Permission.FULL_CONTROL)
                .build();

List<Grant> grantList2 = new ArrayList<>();
grantList2.add(ownerGrant);

AccessControlPolicy acl = AccessControlPolicy.builder()
    .owner(builder -> builder.id(id))
    .grants(grantList2)
    .build();

PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
    .bucket(bucketName)
    .accessControlPolicy(acl)
    .build();

s3.putBucketAcl(putAclReq);

    } catch (S3Exception e) {
        e.printStackTrace();
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス[PutBucketAcl](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## PutBucketCors

次の例は、PutBucketCors を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import java.util.ArrayList;
import java.util.List;
import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <accountId>\s

                Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                accountId - The id of the account that owns the Amazon S3
bucket.

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
```

```
        .build();

        setCorsInformation(s3, bucketName, accountId);
        getBucketCorsInformation(s3, bucketName, accountId);
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
    }

    public static void deleteBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketCors(bucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            GetBucketCorsRequest bucketCorsRequest = GetBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
            List<CORSRule> corsRules = corsResponse.getCorsRules();
            for (CORSRule rule : corsRules) {
                System.out.println("allowOrigins: " + rule.allowedOrigins());
                System.out.println("AllowedMethod: " + rule.allowedMethods());
            }

        } catch (S3Exception e) {

            System.err.println(e.awsErrorDetails().errorMessage());
```

```
        System.exit(1);
    }
}

public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
    List<String> allowMethods = new ArrayList<>();
    allowMethods.add("PUT");
    allowMethods.add("POST");
    allowMethods.add("DELETE");

    List<String> allowOrigins = new ArrayList<>();
    allowOrigins.add("http://example.com");
    try {
        // Define CORS rules.
        CORSRule corsRule = CORSRule.builder()
            .allowedMethods(allowMethods)
            .allowedOrigins(allowOrigins)
            .build();

        List<CORSRule> corsRules = new ArrayList<>();
        corsRules.add(corsRule);
        CORSConfiguration configuration = CORSConfiguration.builder()
            .corsRules(corsRules)
            .build();

        PutBucketCorsRequest putBucketCorsRequest =
PutBucketCorsRequest.builder()
            .bucket(bucketName)
            .corsConfiguration(configuration)
            .expectedBucketOwner(accountId)
            .build();

        s3.putBucketCors(putBucketCorsRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [PutBucketCors](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## PutBucketLifecycleConfiguration

次の例は、PutBucketLifecycleConfiguration を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class LifecycleConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon Simple Storage Service
(Amazon S3) bucket to upload an object into.
                accountId - The id of the account that owns the
Amazon S3 bucket.

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setLifecycleConfig(s3, bucketName, accountId);
        getLifecycleConfig(s3, bucketName, accountId);
        deleteLifecycleConfig(s3, bucketName, accountId);
        System.out.println("You have successfully created, updated, and
deleted a Lifecycle configuration");
        s3.close();
    }

    public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            // Create a rule to archive objects with the
"glacierobjects/" prefix to Amazon
            // S3 Glacier.
            LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
```

```
        .build();

        Transition transition = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
        .days(0)
        .build();

        LifecycleRule rule1 = LifecycleRule.builder()
        .id("Archive immediately rule")
        .filter(ruleFilter)
        .transitions(transition)
        .status(ExpirationStatus.ENABLED)
        .build();

        // Create a second rule.
        Transition transition2 = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
        .days(0)
        .build();

        List<Transition> transitionList = new ArrayList<>();
        transitionList.add(transition2);

        LifecycleRuleFilter ruleFilter2 =
LifecycleRuleFilter.builder()
        .prefix("glacierobjects/")
        .build();

        LifecycleRule rule2 = LifecycleRule.builder()
        .id("Archive and then delete rule")
        .filter(ruleFilter2)
        .transitions(transitionList)
        .status(ExpirationStatus.ENABLED)
        .build();

        // Add the LifecycleRule objects to an ArrayList.
        ArrayList<LifecycleRule> ruleList = new ArrayList<>();
        ruleList.add(rule1);
        ruleList.add(rule2);

        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
```



```
                .rules(ruleList)
                .build();

        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)

        .lifecycleConfiguration(lifecycleConfiguration)
                .expectedBucketOwner(accountId)
                .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Retrieve the configuration and add a new rule.
    public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketLifecycleConfigurationResponse response = s3

        .getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
            List<LifecycleRule> newList = new ArrayList<>();
            List<LifecycleRule> rules = response.rules();
            for (LifecycleRule rule : rules) {
                newList.add(rule);
            }

            // Add a new rule with both a prefix predicate and a tag
predicate.
```

```
        LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                        .prefix("YearlyDocuments/")
                        .build();

        Transition transition = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                .days(3650)
                .build();

        LifecycleRule rule1 = LifecycleRule.builder()
                .id("NewRule")
                .filter(ruleFilter)
                .transitions(transition)
                .status(ExpirationStatus.ENABLED)
                .build();

        // Add the new rule to the list.
        newList.add(rule1);
        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                .rules(newList)
                .build();

        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)

.lifecycleConfiguration(lifecycleConfiguration)
                .expectedBucketOwner(accountId)
                .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the configuration from the Amazon S3 bucket.
```

```
public static void deleteLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
    try {
        DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
DeleteBucketLifecycleRequest
                                .builder()
                                .bucket(bucketName)
                                .expectedBucketOwner(accountId)
                                .build();

        s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [PutBucketLifecycleConfiguration](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutBucketNotificationConfiguration

次の例は、PutBucketNotificationConfiguration を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Event;
import software.amazon.awssdk.services.s3.model.NotificationConfiguration;
import
software.amazon.awssdk.services.s3.model.PutBucketNotificationConfigurationRequest;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.TopicConfiguration;
import java.util.ArrayList;
import java.util.List;

public class SetBucketEventBridgeNotification {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket.\s
                topicArn - The Simple Notification Service topic ARN.\s
                id - An id value used for the topic configuration. This value is
displayed in the AWS Management Console.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String topicArn = args[1];
        String id = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3Client = S3Client.builder()
            .region(region)
            .build();

        setBucketNotification(s3Client, bucketName, topicArn, id);
        s3Client.close();
    }

    public static void setBucketNotification(S3Client s3Client, String bucketName,
String topicArn, String id) {
        try {
            List<Event> events = new ArrayList<>();
            events.add(Event.S3_OBJECT_CREATED_PUT);

            TopicConfiguration config = TopicConfiguration.builder()
                .topicArn(topicArn)
```

```
        .events(events)
        .id(id)
        .build();

    List<TopicConfiguration> topics = new ArrayList<>();
    topics.add(config);

    NotificationConfiguration configuration =
NotificationConfiguration.builder()
        .topicConfigurations(topics)
        .build();

    PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
        .builder()
        .bucket(bucketName)
        .notificationConfiguration(configuration)
        .skipDestinationValidation(true)
        .build();

    // Set the bucket notification configuration.
    s3Client.putBucketNotificationConfiguration(configurationRequest);
    System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス[PutBucketNotificationConfiguration](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutBucketPolicy

次の例は、PutBucketPolicy を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <polFile>

                Where:
                bucketName - The Amazon S3 bucket to set the policy on.
                polFile - A JSON file containing the policy (see the Amazon S3
                Readme for an example).\s
                """;

        if (args.length != 2) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String polFile = args[1];
    String policyText = getBucketPolicyFromFile(polFile);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setPolicy(s3, bucketName, policyText);
    s3.close();
}

public static void setPolicy(S3Client s3, String bucketName, String policyText)
{
    System.out.println("Setting policy:");
    System.out.println("----");
    System.out.println(policyText);
    System.out.println("----");
    System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

    try {
        PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
            .bucket(bucketName)
            .policy(policyText)
            .build();

        s3.putBucketPolicy(policyReq);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("Done!");
}

// Loads a JSON-formatted policy from a file
public static String getBucketPolicyFromFile(String policyFile) {

    StringBuilder fileText = new StringBuilder();
```

```
    try {
        List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
        for (String line : lines) {
            fileText.append(line);
        }

    } catch (IOException e) {
        System.out.format("Problem reading file: \"%s\"", policyFile);
        System.out.println(e.getMessage());
    }

    try {
        final JsonParser parser = new
ObjectMapper().getFactory().createParser(fileText.toString());
        while (parser.nextToken() != null) {
        }

    } catch (IOException jpe) {
        jpe.printStackTrace();
    }
    return fileText.toString();
}
}
```

- API の詳細については、「API リファレンス [PutBucketPolicy](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## PutBucketWebsite

次の例は、PutBucketWebsite を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例  
を見つけて、設定と実行の方法を確認してください。



```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucketName> [indexdoc]\s

            Where:
                bucketName    - The Amazon S3 bucket to set the website
configuration on.\s
                indexdoc    - The index document, ex. 'index.html'
                            If not specified, 'index.html' will be set.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String indexDoc = "index.html";
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setWebsiteConfig(s3, bucketName, indexDoc);
        s3.close();
    }
}
```

```
    }

    public static void setWebsiteConfig(S3Client s3, String bucketName, String
indexDoc) {
        try {
            WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
                .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
                .build();

            PutBucketWebsiteRequest pubWebsiteReq =
PutBucketWebsiteRequest.builder()
                .bucket(bucketName)
                .websiteConfiguration(websiteConfig)
                .build();

            s3.putBucketWebsite(pubWebsiteReq);
            System.out.println("The call was successful");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [PutBucketWebsite](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## PutObject

次の例は、PutObject を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例  
を見つけて、設定と実行の方法を確認してください。

[S3Client](#) を使用してバケットにファイルをアップロードします。

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutObject {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <objectKey> <objectPath>\s

                Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                objectKey - The object to upload (for example, book.pdf).
                objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
```

```
        .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    // This example uses RequestBody.fromFile to avoid loading the whole file into
    // memory.
    public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("x-amz-meta-myVal", "test");
            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .metadata(metadata)
                .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

[S3TransferManager](#) を使用してバケットに [ファイルをアップロード](#) します。 [完全なファイルとテスト](#) を表示します。

```
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();
}
```

[S3Client](#) を使用してオブジェクトをバケットにアップロードし、タグを設定します。

```
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(putObj, RequestBody.fromBytes(getObjectFile(objectPath)));

} catch (S3Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
    try {
        GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
        List<Tag> objTags = getTaggingRes.tagSet();
        for (Tag sinTag : objTags) {
            System.out.println("The tag key is: " + sinTag.key());
            System.out.println("The tag value is: " + sinTag.value());
        }

        // Replace the object's tags with two new tags.
        Tag tag3 = Tag.builder()
            .key("Tag 3")
            .value("This is tag 3")
            .build();

        Tag tag4 = Tag.builder()
            .key("Tag 4")
            .value("This is tag 4")
            .build();

        List<Tag> tags = new ArrayList<>();
        tags.add(tag3);
        tags.add(tag4);
    }
}
```

```
        Tagging updatedTags = Tagging.builder()
            .tagSet(tags)
            .build();

        PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .tagging(updatedTags)
            .build();

        s3.putObjectTagging(taggingRequest1);
        GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
        List<Tag> modTags = getTaggingRes2.tagSet();
        for (Tag sinTag : modTags) {
            System.out.println("The tag key is: " + sinTag.key());
            System.out.println("The tag value is: " + sinTag.value());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Return a byte array.
private static byte[] getObjectFile(String filePath) {
    FileInputStream fileInputStream = null;
    byte[] byteArray = null;

    try {
        File file = new File(filePath);
        byteArray = new byte[(int) file.length()];
        fileInputStream = new FileInputStream(file);
        fileInputStream.read(byteArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
```

```

        e.printStackTrace();
    }
}
}

return byteArray;
}
}

```

[S3Client](#) を使用してオブジェクトをバケットにアップロードし、メタデータを設定します。

```

import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = ""

                Usage:
                <bucketName> <objectKey> <objectPath>\s

                Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                objectKey - The object to upload (for example, book.pdf).
                objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
                """;

```



```
        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
bucketName);
        System.out.println("  in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    // This example uses RequestBody.fromFile to avoid loading the whole file into
    // memory.
    public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .metadata(metadata)
                .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

```
}
```

[S3Client](#) を使用してオブジェクトをバケットにアップロードし、オブジェクトの保持値を設定します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutObjectRetention {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <key> <bucketName>\s

                Where:
                key - The name of the object (for example, book.pdf).\s
                bucketName - The Amazon S3 bucket name that contains the object
                (for example, bucket1).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String key = args[0];
String bucketName = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

setRetentionPeriod(s3, key, bucketName);
s3.close();
}

public static void setRetentionPeriod(S3Client s3, String key, String bucket) {
    try {
        LocalDate localDate = LocalDate.parse("2020-07-17");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        ObjectLockRetention lockRetention = ObjectLockRetention.builder()
            .mode("COMPLIANCE")
            .retainUntilDate(instant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucket)
            .key(key)
            .bypassGovernanceRetention(true)
            .retention(lockRetention)
            .build();

        // To set Retention on an object, the Amazon S3 bucket must support
object
        // locking, otherwise an exception is thrown.
s3.putObjectRetention(retentionRequest);
        System.out.print("An object retention configuration was successfully
placed on the object");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [PutObject](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutObjectLegalHold

次の例は、PutObjectLegalHold を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey, boolean
legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.ON)
            .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.OFF)
            .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .legalHold(legalHold)
        .build();

    getClient().putObjectLegalHold(legalHoldRequest) ;
    System.out.println("Modified legal hold for "+ objectKey +" in "+bucketName
+".");
}
```

```
}
```

- API の詳細については、「API リファレンス [PutObjectLegalHold](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutObjectLockConfiguration

次の例は、PutObjectLockConfiguration を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

バケットのオブジェクトロック設定を指定します。

```
// 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() +
        "'");
    }
}
```

バケットのデフォルトの保存期間を設定します。

```
// Set or modify a retention period on an S3 bucket.
public void modifyBucketDefaultRetention(String bucketName) {
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
        .mfaDelete(MFADelete.DISABLED)
        .status(BucketVersioningStatus.ENABLED)
        .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

    getClient().putBucketVersioning(versioningRequest);
    DefaultRetention retention = DefaultRetention.builder()
        .days(1)
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .build();

    ObjectLockRule lockRule = ObjectLockRule.builder()
        .defaultRetention(retention)
        .build();

    ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
        .objectLockEnabled(ObjectLockEnabled.ENABLED)
        .rule(lockRule)
        .build();
}
```

```
PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .objectLockConfiguration(objectLockConfiguration)
    .build();

getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
System.out.println("Added a default retention to bucket "+bucketName +".");
}
```

- API の詳細については、「API リファレンス [PutObjectLockConfiguration](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutObjectRetention

次の例は、PutObjectRetention を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Set or modify a retention period on an object in an S3 bucket.
public void modifyObjectRetentionPeriod(String bucketName, String objectKey) {
    // Calculate the instant one day from now.
    Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

    // Convert the Instant to a ZonedDateTime object with a specific time zone.
    ZonedDateTime zonedDateTime = futureInstant.atZone(ZoneId.systemDefault());

    // Define a formatter for human-readable output.
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

    // Format the ZonedDateTime object to a human-readable date string.
    String humanReadableDate = formatter.format(zonedDateTime);
}
```

```
// Print the formatted date string.
System.out.println("Formatted Date: " + humanReadableDate);
ObjectLockRetention retention = ObjectLockRetention.builder()
    .mode(ObjectLockRetentionMode.GOVERNANCE)
    .retainUntilDate(futureInstant)
    .build();

PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .retention(retention)
    .build();

getClient().putObjectRetention(retentionRequest);
System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
}
```

- API の詳細については、「API リファレンス[PutObjectRetention](#)」の「」を参照してください。AWS SDK for Java 2.x

## RestoreObject

次の例は、RestoreObject を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
```



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

/*
 * For more information about restoring an object, see "Restoring an archived
 * object" at
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <expectedBucketOwner>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name of an object with a Storage class value
of Glacier.\s
                expectedBucketOwner - The account that owns the bucket (you can
obtain this value from the AWS Management Console).\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
    }
}
```

```
s3.close();
}

public static void restoreS3Object(S3Client s3, String bucketName, String
keyName, String expectedBucketOwner) {
    try {
        RestoreRequest restoreRequest = RestoreRequest.builder()
            .days(10)

.glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
            .build();

        RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
            .expectedBucketOwner(expectedBucketOwner)
            .bucket(bucketName)
            .key(keyName)
            .restoreRequest(restoreRequest)
            .build();

        s3.restoreObject(objectRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [RestoreObject](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## SelectObjectContent

次の例は、SelectObjectContent を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

次の例は、JSON オブジェクトを使用したクエリを示しています。 [完全な例](#) では、CSV オブジェクトの使用も示しています。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

```
public class SelectObjectContentExample {
    static final Logger logger =
    LoggerFactory.getLogger(SelectObjectContentExample.class);
    static final String BUCKET_NAME = "select-object-content-" + UUID.randomUUID();
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
    master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/countries/
    master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
        SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
            System.exit(1);
        } finally {
            SelectObjectContentExample.tearDown();
        }
    }

    EventStreamInfo runSelectObjectContentMethodForJSON() {
        // Set up request parameters.
        final String queryExpression = "select * from s3object[*][*] c where c.area
    < 350000";
        final String fileType = FILE_JSON;

        InputSerialization inputSerialization = InputSerialization.builder()
            .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
            .compressionType(CompressionType.NONE)
            .build();

        OutputSerialization outputSerialization = OutputSerialization.builder()
            .json(JSONOutput.builder().recordDelimiter(null).build())
            .build();

        // Build the SelectObjectContentRequest.
        SelectObjectContentRequest select = SelectObjectContentRequest.builder()
```

```

        .bucket(BUCKET_NAME)
        .key(FILE_JSON)
        .expression(queryExpression)
        .expressionType(ExpressionType.SQL)
        .inputSerialization(inputSerialization)
        .outputSerialization(outputSerialization)
        .build();

    EventStreamInfo eventStreamInfo = new EventStreamInfo();
    // Call the selectObjectContent method with the request and a response
    handler.
    // Supply an EventStreamInfo object to the response handler to gather
    records and information from the response.
    s3AsyncClient.selectObjectContent(select,
    buildResponseHandler(eventStreamInfo)).join();

    // Log out information gathered while processing the response stream.
    long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record ->
        record.split("\n").length
    ).sum();
    logger.info("Total records {}: {}", fileType, recordCount);
    logger.info("Visitor onRecords for fileType {} called {} times", fileType,
    eventStreamInfo.getCountOnRecordsCalled());
    logger.info("Visitor onStats for fileType {}, {}", fileType,
    eventStreamInfo.getStats());
    logger.info("Visitor onContinuations for fileType {}, {}", fileType,
    eventStreamInfo.getCountContinuationEvents());
    return eventStreamInfo;
}

static SelectObjectContentResponseHandler buildResponseHandler(EventStreamInfo
eventStreamInfo) {
    // Use a Visitor to process the response stream. This visitor logs
    information and gathers details while processing.
    final SelectObjectContentResponseHandler.Visitor visitor =
    SelectObjectContentResponseHandler.Visitor.builder()
        .onRecords(r -> {
            logger.info("Record event received.");
            eventStreamInfo.addRecord(r.payload().asUtf8String());
            eventStreamInfo.incrementOnRecordsCalled();
        })
        .onCont(ce -> {
            logger.info("Continuation event received.");
            eventStreamInfo.incrementContinuationEvents();
        })
}

```

```

        })
        .onProgress(pe -> {
            Progress progress = pe.details();
            logger.info("Progress event received:\n bytesScanned:
{} \n bytesProcessed: {} \n bytesReturned: {}",
                progress.bytesScanned(),
                progress.bytesProcessed(),
                progress.bytesReturned());
        })
        .onEnd(ee -> logger.info("End event received. "))
        .onStats(se -> {
            logger.info("Stats event received.");
            eventStreamInfo.addStats(se.details());
        })
        .build();

    // Build the SelectObjectContentResponseHandler with the visitor that
    // processes the stream.
    return SelectObjectContentResponseHandler.builder()
        .subscriber(visitor).build();
}

// The EventStreamInfo class is used to store information gathered while
// processing the response stream.
static class EventStreamInfo {
    private final List<String> records = new ArrayList<>();
    private Integer countOnRecordsCalled = 0;
    private Integer countContinuationEvents = 0;
    private Stats stats;

    void incrementOnRecordsCalled() {
        countOnRecordsCalled++;
    }

    void incrementContinuationEvents() {
        countContinuationEvents++;
    }

    void addRecord(String record) {
        records.add(record);
    }

    void addStats(Stats stats) {
        this.stats = stats;
    }
}

```

```
    }

    public List<String> getRecords() {
        return records;
    }

    public Integer getCountOnRecordsCalled() {
        return countOnRecordsCalled;
    }

    public Integer getCountContinuationEvents() {
        return countContinuationEvents;
    }

    public Stats getStats() {
        return stats;
    }
}
```

- API の詳細については、「API リファレンス [SelectObjectContent](#)」の「」を参照してください。AWS SDK for Java 2.x

## シナリオ

### 署名付き URL を作成する

次のコード例は、Amazon S3 の署名付き URL を作成し、オブジェクトをアップロードする方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

オブジェクトの署名付き URL を生成して、オブジェクトをダウンロードします (GET リクエスト)。

インポート。

```
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;
```

URL を生成する。

```
/* 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();
    }
}

```

次の3つの方法のいずれかを使用して、オブジェクトをダウンロードします。

JDK `URLConnection` (v1.1 以降) クラスを使用してダウンロードを行います。

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setRequestMethod("GET");
        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

JDK HttpClient (v11 以降) クラスを使用してダウンロードを行います。

```
/* Use the JDK HttpClient (since v11) class to do the download. */
public byte[] useHttpClientToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpResponse<InputStream> response = httpClient.send(requestBuilder
            .uri(presignedUrl.toURI())
            .GET()
            .build(),
            HttpResponse.BodyHandlers.ofInputStream());

        IoUtils.copy(response.body(), byteArrayOutputStream);

        logger.info("HTTP response code is " + response.statusCode());
    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}
```

AWS SDK for Java SdkHttpClient クラスを使用してダウンロードを行います。

```
/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToPut(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();
```

```
    HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
        .request(request)
        .build();

    try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
        HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
        response.responseBody().ifPresentOrElse(
            abortableInputStream -> {
                try {
                    IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            },
            () -> logger.error("No response body."));

        logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
    }
} catch (URISyntaxException | IOException e) {
    logger.error(e.getMessage(), e);
}
return byteArrayOutputStream.toByteArray();
}
```

アップロード用の署名付き URL を生成して、ファイルをアップロードします (PUT リクエスト)。

インポート。

```
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;
```

URL を生成する。

```
/* 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();
}
}

```

次の3つの方法のいずれかを使用して、オブジェクトをアップロードします。

JDK `URLConnection` (v1.1以降) クラスを使用してアップロードを行います。

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
public void useURLConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" +
k, v));

        connection.setRequestMethod("PUT");
        OutputStream out = connection.getOutputStream();

        try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
            FileChannel inChannel = file.getChannel()) {
            ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

            while (inChannel.read(buffer) > 0) {
                buffer.flip();
                for (int i = 0; i < buffer.limit(); i++) {
                    out.write(buffer.get());
                }
                buffer.clear();
            }
        }
    }
}

```

```
    }
  } catch (IOException e) {
    logger.error(e.getMessage(), e);
  }

  out.close();
  connection.getResponseCode();
  logger.info("HTTP response code is " + connection.getResponseCode());

} catch (S3Exception | IOException e) {
  logger.error(e.getMessage(), e);
}
}
```

JDK `HttpClient` (v11 以降) クラスを使用してアップロードを行います。

```
/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
  logger.info("Begin [{}] upload", fileToPut.toString());

  HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
  metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

  HttpClient httpClient = HttpClient.newHttpClient();
  try {
    final HttpResponse<Void> response = httpClient.send(requestBuilder
      .uri(new URL(presignedUrlString).toURI())

    .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
      .build(),
      HttpResponse.BodyHandlers.discarding());

    logger.info("HTTP response code is " + response.statusCode());

  } catch (URISyntaxException | InterruptedException | IOException e) {
    logger.error(e.getMessage(), e);
  }
}
```

AWS for Java V2 `SdkHttpClient` クラスを使用してアップロードを実行します。

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

## 不完全なマルチパートアップロードを削除する

次のコード例は、未完了の Amazon S3 マルチパートアップロードを削除または停止する方法を示しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

何らかの理由で進行中または不完全なマルチパートアップロードを停止するには、次の例に示すように、リストアップロードを取得してから削除できます。

```
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);
        }
    }
}
```

日付の前後に開始された不完全なマルチパートアップロードを削除するには、次の例に示すように、特定の時点に基づいてマルチパートアップロードを選択的に削除できます。



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

マルチパートアップロードの開始後にアップロード ID にアクセスできる場合は、ID を使用して進行中のアップロードを削除できます。

```
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);
        }
    }
}
```

不完全なマルチパートアップロードを特定の日数だけ古いマルチパートアップロードを一貫して削除するには、バケットのバケットライフサイクル設定を設定します。次の例は、7 日以上経過した未完了のアップロードを削除するルールを作成する方法を示しています。

```
static void abortMultipartUploadsUsingLifecycleConfig() {
    Collection<LifecycleRule> lifeCycleRules = List.of(LifecycleRule.builder()
        .abortIncompleteMultipartUpload(b -> b.
            daysAfterInitiation(7))
        .status("Enabled")
        .filter(SdkBuilder::build) // Filter element is required.
        .build());

    // If the action is successful, the service sends back an HTTP 200 response
with an empty HTTP body.
    PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
        .bucket(bucketName)
        .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
    } else {
        logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
    }
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [AbortMultipartUpload](#)
  - [ListMultipartUploads](#)
  - [PutBucketLifecycleConfiguration](#)

## ローカルディレクトリへのオブジェクトのダウンロード

次のコード例は、Amazon Simple Storage Service (Amazon S3) バケットからすべてのオブジェクトをローカルディレクトリにダウンロードする方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

[S3TransferManager](#) を使用して、同じ [S3 バケット内のすべての S3 オブジェクトをダウンロード](#) します。 [S3 完全なファイル](#) と [テスト](#) を表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
        URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
            .destination(Paths.get(destinationPathURI))
            .bucket(bucketName)
            .build());
```

```
CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}
```

- API の詳細については、「API リファレンス [DownloadDirectory](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## バケットとオブジェクトの使用を開始する

次のコードサンプルは、以下の操作方法を示しています。

- バケットを作成し、そこにファイルをアップロードします。
- バケットからオブジェクトをダウンロードします。
- バケット内のサブフォルダにオブジェクトをコピーします。
- バケット内のオブジェクトを一覧表示します。
- バケットオブジェクトとバケットを削除します。

## SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
* This Java code example performs the following tasks:  
*  
* 1. Creates an Amazon S3 bucket.  
* 2. Uploads an object to the bucket.  
* 3. Downloads the object to another local file.  
* 4. Uploads an object using multipart upload.  
* 5. List all objects located in the Amazon S3 bucket.  
* 6. Copies the object to another Amazon S3 bucket.  
* 7. Deletes the object from the Amazon S3 bucket.  
* 8. Deletes the Amazon S3 bucket.  
*/
```

```
public class S3Scenario {  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
  
    public static void main(String[] args) throws IOException {  
        final String usage = ""  
  
            Usage:  
            <bucketName> <key> <objectPath> <savePath> <toBucket>  
  
            Where:  
            bucketName - The Amazon S3 bucket to create.  
            key - The key to use.  
            objectPath - The path where the file is located (for example,  
C:/AWS/book2.pdf).  
            savePath - The path where the file is saved after it's  
downloaded (for example, C:/AWS/book2.pdf).  
            toBucket - An Amazon S3 bucket to where an object is copied to  
(for example, C:/AWS/book2.pdf).\s  
            """;  
  
        if (args.length != 5) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        String bucketName = args[0];  
        String key = args[1];  
        String objectPath = args[2];  
        String savePath = args[3];  
        String toBucket = args[4];  
        Region region = Region.US_EAST_1;  
        S3Client s3 = S3Client.builder()
```

```
        .region(region)
        .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon S3 example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an Amazon S3 bucket.");
createBucket(s3, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Update a local file to the Amazon S3 bucket.");
uploadLocalFile(s3, bucketName, key, objectPath);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Download the object to another local file.");
getObjectBytes(s3, bucketName, key, savePath);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Perform a multipart upload.");
String multipartKey = "multiPartKey";
multipartUpload(s3, toBucket, multipartKey);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. List all objects located in the Amazon S3 bucket.");
listAllObjects(s3, bucketName);
anotherListExample(s3, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Copy the object to another Amazon S3 bucket.");
copyBucketObject(s3, bucketName, key, toBucket);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the object from the Amazon S3 bucket.");
deleteObjectFromBucket(s3, bucketName, key);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("8. Delete the Amazon S3 bucket.");
deleteBucket(s3, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("All Amazon S3 operations were successfully performed");
System.out.println(DASHES);
s3.close();
}

// Create a bucket by using a S3Waiter object.
public static void createBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteBucket(S3Client client, String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    client.deleteBucket(deleteBucketRequest);
    System.out.println(bucket + " was deleted.");
}
```

```
/**
 * Upload an object in parts.
 */
public static void multipartUpload(S3Client s3, String bucketName, String key) {
    int mB = 1024 * 1024;
    // First create a multipart upload and get the upload id.
    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(key)
    .build();

    CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
    String uploadId = response.uploadId();
    System.out.println(uploadId);

    // Upload all the different parts of the object.
    UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .partNumber(1).build();

    String etag1 = s3.uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
    .eTag();
    CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

    UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
    .uploadId(uploadId)
    .partNumber(2).build();

    String etag2 = s3.uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB)))
    .eTag();
    CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

    // Call completeMultipartUpload operation to tell S3 to merge all uploaded
    // parts and finish the multipart operation.
```



```
CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
    .parts(part1, part2)
    .build();

CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .multipartUpload(completedMultipartUpload)
    .build();

s3.completeMultipartUpload(completeMultipartUploadRequest);
}

private static ByteBuffer getRandomByteBuffer(int size) {
    byte[] b = new byte[size];
    new Random().nextBytes(b);
    return ByteBuffer.wrap(b);
}

public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
        byte[] data = objectBytes.asByteArray();

        // Write the data to a local file.
        File myFile = new File(path);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
        System.out.println("Successfully obtained bytes from an S3 object");
        os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    }
}
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void uploadLocalFile(S3Client s3, String bucketName, String key,
String objectPath) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        s3.putObject(objectRequest, RequestBody.fromFile(new File(objectPath)));
    }

    public static void listAllObjects(S3Client s3, String bucketName) {
        ListObjectsV2Request listObjectsReqManual = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        boolean done = false;
        while (!done) {
            ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
            for (S3Object content : listObjResponse.contents()) {
                System.out.println(content.key());
            }

            if (listObjResponse.nextContinuationToken() == null) {
                done = true;
            }

            listObjectsReqManual = listObjectsReqManual.toBuilder()
                .continuationToken(listObjResponse.nextContinuationToken())
                .build();
        }
    }

    public static void anotherListExample(S3Client s3, String bucketName) {
        ListObjectsV2Request listReq = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
```

```
        .build());

    ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);

    // Process response pages.
    listRes.stream()
        .flatMap(r -> r.contents().stream())
        .forEach(content -> System.out.println(" Key: " + content.key() + "
size = " + content.size()));

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

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

public static void deleteObjectFromBucket(S3Client s3, String bucketName, String
key) {
    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    s3.deleteObject(deleteObjectRequest);
    System.out.println(key + " was deleted");
}

public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
    String encodedUrl = null;
    try {
        encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
StandardCharsets.UTF_8.toString());
    } catch (UnsupportedEncodingException e) {
        System.out.println("URL could not be encoded: " + e.getMessage());
    }
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .copySource(encodedUrl)
        .destinationBucket(toBucket)
```

```
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        System.out.println("The " + objectKey + " was copied to " + toBucket);
        return copyRes.copyObjectResult().toString();


    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjectsV2](#)
  - [PutObject](#)

オブジェクトのリーガルホールド設定を取得する

次のコード例は、S3 バケットのリーガルホールド設定を取得する方法を示しています。

SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
            ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();

    } catch (S3Exception ex) {
        System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
        "'");
    }

    return null;
}
```

- API の詳細については、「API リファレンス[GetObjectLegalHold](#)」の「」を参照してください。AWS SDK for Java 2.x

## Amazon S3 オブジェクトをロックする

次のコード例は、S3 オブジェクトロック機能を実行する方法を示しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## Amazon S3 オブジェクトロック機能を示すインタラクティブなシナリオを実行します。

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

/*
Before running this Java V2 code example, set up your development
environment, including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html

This Java example performs the following tasks:
  1. Create test Amazon Simple Storage Service (S3) buckets with different lock
  policies.
  2. Upload sample objects to each bucket.
  3. Set some Legal Hold and Retention Periods on objects and buckets.
  4. Investigate lock policies by viewing settings or attempting to delete or
  overwrite objects.
  5. Clean up objects and buckets.
*/
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        // Get the current date and time to ensure bucket name is unique.
        LocalDateTime currentTime = LocalDateTime.now();

        // Format the date and time as a string.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMddHHmmss");
        String timeStamp = currentTime.format(formatter);
```

```
s3LockActions = new S3LockActions();
bucketName = "bucket"+timeStamp;
Scanner scanner = new Scanner(System.in);

System.out.println(DASHES);
System.out.println("Welcome to the Amazon Simple Storage Service (S3) Object
Locking Workflow Scenario.");
System.out.println("Press Enter to continue...");
scanner.nextLine();
configurationSetup();
System.out.println(DASHES);

System.out.println(DASHES);
setup();
System.out.println("Setup is complete. Press Enter to continue...");
scanner.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Lets present the user with choices.");
System.out.println("Press Enter to continue...");
scanner.nextLine();
demoActionChoices() ;
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Would you like to clean up the resources? (y/n)");
String delAns = scanner.nextLine().trim();
if (delAns.equalsIgnoreCase("y")) {
    cleanup();
    System.out.println("Clean up is complete.");
}

System.out.println("Press Enter to continue...");
scanner.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Amazon S3 Object Locking Workflow is complete.");
System.out.println(DASHES);
}

// Present the user with the demo action choices.
```

```
public static void demoActionChoices() {
    String[] choices = {
        "List all files in buckets.",
        "Attempt to delete a file.",
        "Attempt to delete a file with retention period bypass.",
        "Attempt to overwrite a file.",
        "View the object and bucket retention settings for a file.",
        "View the legal hold settings for a file.",
        "Finish the workflow."
    };

    int choice = 0;
    while (true) {
        System.out.println(DASHES);
        choice = getChoiceResponse("Explore the S3 locking features by selecting
one of the following choices:", choices);
        System.out.println(DASHES);
        System.out.println("You selected "+choices[choice]);
        switch (choice) {
            case 0 -> {
                s3LockActions.listBucketsAndObjects(bucketNames, true);
            }

            case 1 -> {
                System.out.println("Enter the number of the object to delete:");
                List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                String[] fileKeysArray = fileKeys.toArray(new String[0]);
                int fileChoice = getChoiceResponse(null, fileKeysArray);
                String objectKey = fileKeys.get(fileChoice);
                String bucketName = allFiles.get(fileChoice).getBucketName();
                String version = allFiles.get(fileChoice).getVersion();
                s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
            }

            case 2 -> {
                System.out.println("Enter the number of the object to delete:");
                List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
```



```
String[] fileKeysArray = fileKeys.toArray(new String[0]);
int fileChoice = getChoiceResponse(null, fileKeysArray);
String objectKey = fileKeys.get(fileChoice);
String bucketName = allFiles.get(fileChoice).getBucketName();
String version = allFiles.get(fileChoice).getVersion();
s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
    }

    case 3 -> {
        System.out.println("Enter the number of the object to
overwrite:");

        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();

        // Attempt to overwrite the file.
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
            writer.write("This is a modified text.");

        } catch (IOException e) {
            e.printStackTrace();
        }
        s3LockActions.uploadFile(bucketName, objectKey, objectKey);
    }

    case 4 -> {
        System.out.println("Enter the number of the object to
overwrite:");

        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        s3LockActions.getObjectRetention(bucketName, objectKey);
```

```

    }

    case 5 -> {
        System.out.println("Enter the number of the object to view:");
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        s3LockActions.getObjectLegalHold(bucketName, objectKey);
        s3LockActions.getBucketObjectLockConfiguration(bucketName);
    }

    case 6 -> {
        System.out.println("Exiting the workflow...");
        return;
    }

    default -> {
        System.out.println("Invalid choice. Please select again.");
    }
}
}

// Clean up the resources from the scenario.
private static void cleanup() {
    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
    for (S3InfoObject fileInfo : allFiles) {
        String bucketName = fileInfo.getBucketName();
        String key = fileInfo.getKeyName();
        String version = fileInfo.getVersion();
        if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
            ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
            if (legalHold != null) {
                String holdStatus = legalHold.status().name();
                System.out.println(holdStatus);
                if (holdStatus.compareTo("ON") == 0) {

```

```
        s3LockActions.modifyObjectLegalHold(bucketName, key, false);
    }
}
// Check for a retention period.
ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
boolean hasRetentionPeriod ;
hasRetentionPeriod = retention != null;
s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

    } else {
        System.out.println(bucketName + " objects do not have a legal lock");
s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
    }
}

// Delete the buckets.
System.out.println("Delete "+bucketName);
for (String bucket : bucketNames){
    s3LockActions.deleteBucketByName(bucket);
}
}

private static void setup() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("""
        For this workflow, we will use the AWS SDK for Java to create
several S3
        buckets and files to demonstrate working with S3 locking features.
        """);

    System.out.println("S3 buckets can be created either with or without object
lock enabled.");
    System.out.println("Press Enter to continue...");
    scanner.nextLine();

    // Create three S3 buckets.
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
    s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();
}
```

```
        System.out.println("Bucket "+bucketNames.get(2) +" will be configured to use
object locking with a default retention period.");
        s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
        s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        // Upload some files to the buckets.
        System.out.println("Now let's add some test files:");
        String fileName = "exampleFile.txt";
        int fileCount = 2;
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
            writer.write("This is a sample file for uploading to a bucket.");

        } catch (IOException e) {
            e.printStackTrace();
        }

        for (String bucketName : bucketNames){
            for (int i = 0; i < fileCount; i++) {
                // Get the file name without extension.
                String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
                int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
                if (extensionIndex > 0) {
                    fileNameWithoutExtension = fileNameWithoutExtension.substring(0,
extensionIndex);
                }

                // Create the numbered file names.
                String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
                fileNames.add(numberedFileName);
                s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
            }
        }
    }
}
```

```

String question = null;
System.out.print("Press Enter to continue...");
scanner.nextLine();
System.out.println("Now we can set some object lock policies on individual
files:");
for (String bucketName : bucketNames) {
    for (int i = 0; i < fileNames.size(); i++){

        // No modifications to the objects in the first bucket.
        if (!bucketName.equals(bucketNames.get(0))) {
            String exampleFileName = fileNames.get(i);
            switch (i) {
                case 0 -> {
                    question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
                    System.out.println(question);
                    String ans = scanner.nextLine().trim();
                    if (ans.equalsIgnoreCase("y")) {
                        System.out.println("**** You have selected to put a
legal hold " + exampleFileName);

                            // Set a legal hold.
                            s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
                                }
                            }
                        case 1 -> {
                            ""
                                Would you like to add a 1 day Governance retention
period to %s in %s (y/n)?

                                    Reminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.
                                        "".formatted(exampleFileName, bucketName);
                                            System.out.println(question);
                                                String ans2 = scanner.nextLine().trim();
                                                    if (ans2.equalsIgnoreCase("y")) {

s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```
    }  
  }  
  
  // Get file extension.  
  private static String getFileExtension(String fileName) {  
    int dotIndex = fileName.lastIndexOf('.');  
    if (dotIndex > 0) {  
      return fileName.substring(dotIndex);  
    }  
    return "";  
  }  
  
  public static void configurationSetup() {  
    String noLockBucketName = bucketName + "-no-lock";  
    String lockEnabledBucketName = bucketName + "-lock-enabled";  
    String retentionAfterCreationBucketName = bucketName + "-retention-after-  
creation";  
    bucketNames.add(noLockBucketName);  
    bucketNames.add(lockEnabledBucketName);  
    bucketNames.add(retentionAfterCreationBucketName);  
  }  
  
  public static int getChoiceResponse(String question, String[] choices) {  
    Scanner scanner = new Scanner(System.in);  
    if (question != null) {  
      System.out.println(question);  
      for (int i = 0; i < choices.length; i++) {  
        System.out.println("\t" + (i + 1) + ". " + choices[i]);  
      }  
    }  
  
    int choiceNumber = 0;  
    while (choiceNumber < 1 || choiceNumber > choices.length) {  
      String choice = scanner.nextLine();  
      try {  
        choiceNumber = Integer.parseInt(choice);  
      } catch (NumberFormatException e) {  
        System.out.println("Invalid choice. Please enter a valid number.");  
      }  
    }  
  
    return choiceNumber - 1;  
  }  
}
```

### S3 関数のラッパークラス。

```
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.waitForBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }

    public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
        Boolean interactive) {
        AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
        return bucketNames.stream()
            .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
                .map(version -> {
                    S3InfoObject s3InfoObject = new S3InfoObject();
                    s3InfoObject.setBucketName(bucketName);
                    s3InfoObject.setVersion(version.versionId());
                    s3InfoObject.setKeyName(version.key());
                    return s3InfoObject;
                })
            .peek(s3InfoObject -> {
                int i = counter.incrementAndGet(); // Increment and get the updated
value.
                if (interactive) {
                    System.out.println(i + ": " + s3InfoObject.getKeyName());
                    System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
                    System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
                }
            })
            .collect(Collectors.toList());
    }

    public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
        ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
            .bucket(bucketName)
            .build();

        return getClient().listObjectVersions(versionsRequest);
    }

    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {

```

```
VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
    .mfaDelete(MFADelete.DISABLED)
    .status(BucketVersioningStatus.ENABLED)
    .build();

PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
    .bucket(bucketName)
    .versioningConfiguration(versioningConfiguration)
    .build();

getClient().putBucketVersioning(versioningRequest);
DefaultRetention retention = DefaultRetention.builder()
    .days(1)
    .mode(ObjectLockRetentionMode.GOVERNANCE)
    .build();

ObjectLockRule lockRule = ObjectLockRule.builder()
    .defaultRetention(retention)
    .build();

ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
    .objectLockEnabled(ObjectLockEnabled.ENABLED)
    .rule(lockRule)
    .build();

PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .objectLockConfiguration(objectLockConfiguration)
    .build();

getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
System.out.println("Added a default retention to bucket "+bucketName +".");
}

// Enable object lock on an existing bucket.
public void enableObjectLockOnBucket(String bucketName) {
    try {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
```

```
        .build();

        PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

        // Enable versioning on the bucket.
        getClient().putBucketVersioning(putBucketVersioningRequest);
        PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .objectLockConfiguration(ObjectLockConfiguration.builder()
                .objectLockEnabled(ObjectLockEnabled.ENABLED)
                .build())
        .build();

        getClient().putObjectLockConfiguration(request);
        System.out.println("Successfully enabled object lock on "+bucketName);

    } catch (S3Exception ex) {
        System.out.println("Error modifying object lock: '" + ex.getMessage() +
        """);
    }
}

public void uploadFile(String bucketName, String objectName, String filePath) {
    Path file = Paths.get(filePath);
    PutObjectRequest request = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(objectName)
        .checksumAlgorithm(ChecksumAlgorithm.SHA256)
        .build();

    PutObjectResponse response = getClient().putObject(request, file);
    if (response != null) {
        System.out.println("\tSuccessfully uploaded " + objectName + " to " +
bucketName + ".");
    } else {
        System.out.println("\tCould not upload " + objectName + " to " +
bucketName + ".");
    }
}
}
```

```
// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey, boolean
legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.ON)
            .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.OFF)
            .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .legalHold(legalHold)
    .build();

    getClient().putObjectLegalHold(legalHoldRequest) ;
    System.out.println("Modified legal hold for " + objectKey + " in " + bucketName
+ ".");
}

// Delete an object from a specific bucket.
public void deleteObjectFromBucket(String bucketName, String objectKey, boolean
hasRetention, String versionId) {
    try {
        DeleteObjectRequest objectRequest;
        if (hasRetention) {
            objectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .versionId(versionId)
                .bypassGovernanceRetention(true)
                .build();
        } else {
            objectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .versionId(versionId)
```

```
        .build();
    }

    getClient().deleteObject(objectRequest) ;
    System.out.println("The object was successfully deleted");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}
}

// Get the retention period for an S3 object.
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {
        GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

        GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
        System.out.println("Object retention for "+key+" in "+ bucketName +":
" + response.retention().mode() +" until "+ response.retention().retainUntilDate()
+".");
        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}

public void deleteBucketByName(String bucketName) {
    try {
        DeleteBucketRequest request = DeleteBucketRequest.builder()
        .bucket(bucketName)
        .build();

        getClient().deleteBucket(request);
        System.out.println(bucketName +" was deleted.");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
    }  
  }  
  
  // Get the object lock configuration details for an S3 bucket.  
  public void getBucketObjectLockConfiguration(String bucketName) {  
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =  
    GetObjectLockConfigurationRequest.builder()  
      .bucket(bucketName)  
      .build();  
  
    GetObjectLockConfigurationResponse response =  
    getClient().getObjectLockConfiguration(objectLockConfigurationRequest);  
    System.out.println("Bucket object lock config for "+bucketName+": ");  
    System.out.println("\tEnabled:  
"+response.getObjectLockConfiguration().getObjectLockEnabled());  
    System.out.println("\tRule: "+  
response.getObjectLockConfiguration().rule().defaultRetention());  
  }  
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [GetObjectLegalHold](#)
  - [GetObjectLockConfiguration](#)
  - [GetObjectRetention](#)
  - [PutObjectLegalHold](#)
  - [PutObjectLockConfiguration](#)
  - [PutObjectRetention](#)

## URI を解析する

次のコード例は、Amazon S3 の URI を解析してバケット名やオブジェクトキーなどの重要なコンポーネントを抽出する方法を示しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

[S3Uri](#) クラスを使用して Amazon S3 の URI を解析します。

```
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);
    }
}
}

```

## マルチパートアップロードの実行

次のコード例は、Amazon S3 オブジェクトへのマルチパートアップロードを実行する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

コード例では、次のインポートを使用します。

```

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;
```

[AWS CRT ベースの S3 クライアント](#) 上にある [S3 Transfer Manager](#) を使用すると、コンテンツのサイズがしきい値を超えたときにマルチパートアップロードを透過的に実行できます。デフォルトのしきい値は 8 MB です。

```
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();
}
```

```
}
```

[S3Client API](#) を使用してマルチパートアップロードを実行します。

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

マルチパートアップロードを実行するには、マルチパートサポートを有効にした [S3AsyncClient API](#) を使用します。

```
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.");
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。

- [CompleteMultipartUpload](#)
- [CreateMultipartUpload](#)
- [UploadPart](#)

## アップロードとダウンロードを追跡する

次のコード例は、Amazon S3 オブジェクトのアップロードまたはダウンロードを追跡する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

ファイルのアップロードの進行状況を追跡します。

```
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!
*/
}

```

ファイルのダウンロードの進捗状況を追跡します。

```

    public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                                String key, String downloadedFilePath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
            .getObjectRequest(b -> b.bucket(bucketName).key(key))
            .addTransferListener(LoggingTransferListener.create()) // Add
listener.
            .destination(Paths.get(downloadedFilePath))
            .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
    /*

```

The SDK provides a `LoggingTransferListener` implementation of the `TransferListener` interface.

You can also implement the interface to provide your own logic.

Configure log4j2 with settings such as the following.

```
<Configuration status="WARN">
  <Appenders>
    <Console name="AlignedConsoleAppender" target="SYSTEM_OUT">
      <PatternLayout pattern="%m%n"/>
    </Console>
  </Appenders>

  <Loggers>
    <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
      <AppenderRef ref="AlignedConsoleAppender"/>
    </logger>
  </Loggers>
</Configuration>
```

Log4j2 logs the progress. The following is example output for a 21.3 MB file download.

```
Transfer initiated...
|=====| 39.4%
|=====| 78.8%
|=====| 100.0%
Transfer complete!

  */
}
```

- APIの詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [GetObject](#)
  - [PutObject](#)

## バケットへのディレクトリのアップロード

以下のコード例では、ローカルディレクトリを再帰的に Amazon Simple Storage Service (Amazon S3) バケットにアップロードする方法を示します。



## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

[S3TransferManager](#) を使用して [ローカルディレクトリ](#) をアップロードします。 [完全なファイルとテスト](#) を表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public Integer uploadDirectory(S3TransferManager transferManager,
        URI sourceDirectory, String bucketName) {
        DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
        .source(Paths.get(sourceDirectory))
        .bucket(bucketName)
        .build());

        CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
            .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

- API の詳細については、「API リファレンス [UploadDirectory](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## 大きなファイルをアップロードまたはダウンロードする

次のコード例は、Amazon S3 との間で大きなファイルをアップロードまたはダウンロードする方法を示しています。

詳細については、「[マルチパートアップロードを使用したオブジェクトのアップロード](#)」を参照してください。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

S3 を使用して S3 バケットとの間でファイルを転送する関数を呼び出します  
S3TransferManager。

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
        .destination(Paths.get(destinationPathURI))
        .bucket(bucketName)
        .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}
```

ローカルディレクトリ全体をアップロードします。

```
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();
}
```

1つのファイルをアップロードします。

```
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();
}
```

サイズが不明なストリームをアップロードする

次のコード例は、サイズが不明なストリームを Amazon S3 オブジェクトにアップロードする方法を示しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

### [AWS CRT ベースの S3 クライアント](#)を使用します。

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

/**
 * @param s3CrtAsyncClient - To upload content from a stream of unknown size,
 * use the AWS CRT-based S3 client. For more information, see
 * https://docs.aws.amazon.com/sdk-for-java/latest/
 * developer-guide/crt-based-s3-client.html.
 * @param bucketName - The name of the bucket.
 * @param key - The name of the object.
 * @return software.amazon.awssdk.services.s3.model.PutObjectResponse - Returns
 * metadata pertaining to the put object operation.
 */
public PutObjectResponse putObjectFromStream(S3AsyncClient s3CrtAsyncClient,
String bucketName, String key) {

    BlockingInputStreamAsyncRequestBody body =
        AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
stream will be provided later.

    CompletableFuture<PutObjectResponse> responseFuture =
        s3CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
body);
```

```
        // AsyncExampleUtils.randomString() returns a random string up to 100
        characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
            randomString.length());

        // Provide the stream of data to be uploaded.
        body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

        PutObjectResponse response = responseFuture.join(); // Wait for the
        response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        return response;
    }
}
```

[Amazon S3 Transfer Manager](#) を使用します。

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;

import java.io.ByteArrayInputStream;
import java.util.UUID;

/**
 * @param transferManager - To upload content from a stream of unknown size, use
 * the S3TransferManager based on the AWS CRT-based S3 client.
 *
 * For more information, see https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/transfer-manager.html.
 * @param bucketName - The name of the bucket.
 * @param key - The name of the object.
 * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
 * result of the completed upload.
 */
```

```
public CompletedUpload uploadStream(S3TransferManager transferManager, String
bucketName, String key) {

    BlockingInputStreamAsyncRequestBody body =
        AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
stream will be provided later.

    Upload upload = transferManager.upload(builder -> builder
        .requestBody(body)
        .putObjectRequest(req -> req.bucket(bucketName).key(key))
        .build());

    // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
randomString.length());

    // Provide the stream of data to be uploaded.
    body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

    return upload.completionFuture().join();
}
}
```

## チェックサムの使用

次のコード例は、チェックサムを使用して Amazon S3 オブジェクトを操作する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

コード例では、以下のインポートのサブセットを使用しています。

```
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;
```

[PutObjectRequest をビルドする](#) ときに、putObject メソッドのチェックサムアルゴリズムを指定します。

```
public void putObjectWithChecksum() {
    s3Client.putObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(ChecksumAlgorithm.CRC32),
```

```
        RequestBody.fromString("This is a test"));
    }
```

を構築するときに、`getObject`メソッドのチェックサムを確認します。 [GetObjectRequest](#)

```
public GetObjectResponse getObjectWithChecksum() {
    return s3Client.getObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
        .response();
}
```

[PutObjectRequest](#) をビルドするときに、`putObject` メソッドのチェックサムを事前に計算します。

```
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)));
}
```

[AWS CRT ベースの S3 クライアント](#) 上にある [S3 Transfer Manager](#) を使用して、コンテンツのサイズがしきい値を超えたときにマルチパートアップロードを透過的に実行します。デフォルトのしきい値は 8 MB です。

SDK で使用するチェックサムアルゴリズムを指定できます。デフォルトでは、SDK は CRC32 アルゴリズムを使用します。

```
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();
}

```

[S3Client API](#) または ([S3AsyncClient API](#)) を使用して、マルチパートアップロードを実行します。追加のチェックサムを指定する場合は、アップロードの開始時に使用するアルゴリズムを指定する必要があります。また、パートリクエストごとにアルゴリズムを指定し、アップロード後にパートごとに計算されたチェックサムを指定する必要があります。

```

public void multipartUploadWithChecksumS3Client(String filePath) {
    ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(algorithm)); // Checksum specified on initiation.
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        long position = 0;
        while (position < fileSize) {
            file.seek(position);
            long read = file.getChannel().read(bb);

            bb.flip(); // Swap position and limit before reading from the
buffer.

            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)

```

```
        .uploadId(uploadId)
        .checksumAlgorithm(algorithm) // Checksum specified on each
part.
        .partNumber(partNumber)
        .build();

UploadPartResponse partResponse = s3Client.uploadPart(
    uploadPartRequest,
    RequestBody.fromByteBuffer(bb));

CompletedPart part = CompletedPart.builder()
    .partNumber(partNumber)
    .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.
    .eTag(partResponse.eTag())
    .build();
completedParts.add(part);

bb.clear();
position += read;
partNumber++;
    }
} catch (IOException e) {
    System.err.println(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [CompleteMultipartUpload](#)
  - [CreateMultipartUpload](#)
  - [UploadPart](#)

## サーバーレスサンプル

### Amazon S3 トリガーから Lambda 関数を呼び出す

次のコード例は、S3 バケットにオブジェクトをアップロードすることによってトリガーされるイベントを受け取る Lambda 関数を実装する方法を示しています。この関数は、イベントパラメータから S3 バケット名とオブジェクトキーを取得し、Amazon S3 API を呼び出してオブジェクトのコンテンツタイプを取得してログに記録します。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [サーバーレスサンプル](#) リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Java を使用して Lambda で S3 イベントを消費します。

```
// 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.S3EventNotificationRecord;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
```

```
String srcBucket = record.getS3().getBucket().getName();
String srcKey = record.getS3().getObject().getUrlDecodedKey();

S3Client s3Client = S3Client.builder().build();
HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());

return "Ok";
} catch (Exception e) {
throw new RuntimeException(e);
}
}

private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
.bucket(bucket)
.key(key)
.build();
return s3Client.headObject(headObjectRequest);
}
}
```

## SDK for Java 2.x を使用した S3 Glacier の例

次のコード例は、S3 Glacier AWS SDK for Java 2.x でを使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### トピック

## • [アクション](#)

### アクション

#### CreateVault

次の例は、CreateVault を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.CreateVaultRequest;
import software.amazon.awssdk.services.glacier.model.CreateVaultResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateVault {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <vaultName>

                Where:
                    vaultName - The name of the vault to create.

        """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void createGlacierVault(GlacierClient glacier, String vaultName) {
        try {
            CreateVaultRequest vaultRequest = CreateVaultRequest.builder()
                .vaultName(vaultName)
                .build();

            CreateVaultResponse createVaultResult =
glacier.createVault(vaultRequest);
            System.out.println("The URI of the new vault is " +
createVaultResult.location());

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [CreateVault](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteArchive

次の例は、DeleteArchive を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「API リファレンス [DeleteArchive](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteVault

次の例は、DeleteVault を使用する方法を説明しています。



## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「API リファレンス [DeleteVault](#)」の「」を参照してください。AWS SDK for Java 2.x

## InitiateJob

次の例は、InitiateJob を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

ポールトインベントリを取得します。

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.JobParameters;
```

```
import software.amazon.awssdk.services.glacier.model.InitiateJobResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import software.amazon.awssdk.services.glacier.model.InitiateJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobResponse;
import software.amazon.awssdk.services.glacier.model.GetJobOutputRequest;
import software.amazon.awssdk.services.glacier.model.GetJobOutputResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ArchiveDownload {
    public static void main(String[] args) {

        final String usage = ""

            Usage:    <vaultName> <accountId> <path>

            Where:
                vaultName - The name of the vault.
                accountId - The account ID value.
                path - The path where the file is written to.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        String accountId = args[1];
        String path = args[2];
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();
```

```
String jobNum = createJob(glacier, vaultName, accountId);
checkJob(glacier, jobNum, vaultName, accountId, path);
glacier.close();
}

public static String createJob(GlacierClient glacier, String vaultName, String
accountId) {
    try {
        JobParameters job = JobParameters.builder()
            .type("inventory-retrieval")
            .build();

        InitiateJobRequest initJob = InitiateJobRequest.builder()
            .jobParameters(job)
            .accountId(accountId)
            .vaultName(vaultName)
            .build();

        InitiateJobResponse response = glacier.initiateJob(initJob);
        System.out.println("The job ID is: " + response.jobId());
        System.out.println("The relative URI path of the job is: " +
response.location());
        return response.jobId();

    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

// Poll S3 Glacier = Polling a Job may take 4-6 hours according to the
// Documentation.
public static void checkJob(GlacierClient glacier, String jobId, String name,
String account, String path) {
    try {
        boolean finished = false;
        String jobStatus;
        int yy = 0;

        while (!finished) {
            DescribeJobRequest jobRequest = DescribeJobRequest.builder()
```

```
        .jobId(jobId)
        .accountId(account)
        .vaultName(name)
        .build();

DescribeJobResponse response = glacier.describeJob(jobRequest);
jobStatus = response.statusCodeAsString();

if (jobStatus.compareTo("Succeeded") == 0)
    finished = true;
else {
    System.out.println(yy + " status is: " + jobStatus);
    Thread.sleep(1000);
}
yy++;
}

System.out.println("Job has Succeeded");
GetJobOutputRequest jobOutputRequest = GetJobOutputRequest.builder()
    .jobId(jobId)
    .vaultName(name)
    .accountId(account)
    .build();

ResponseBytes<GetJobOutputResponse> objectBytes =
glacier.getJobOutputAsBytes(jobOutputRequest);
// Write the data to a local file.
byte[] data = objectBytes.asByteArray();
File myFile = new File(path);
OutputStream os = new FileOutputStream(myFile);
os.write(data);
System.out.println("Successfully obtained bytes from a Glacier vault");
os.close();

} catch (GlacierException | InterruptedException | IOException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
}
```

- API の詳細については、「API リファレンス [InitiateJob](#)」の「 」を参照してください。AWS SDK for Java 2.x

## ListVaults

次の例は、ListVaults を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「 」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.model.ListVaultsRequest;
import software.amazon.awssdk.services.glacier.model.ListVaultsResponse;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DescribeVaultOutput;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListVaults {
    public static void main(String[] args) {
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllVault(glacier);
        glacier.close();
    }

    public static void listAllVault(GlacierClient glacier) {
```

```
boolean listComplete = false;
String newMarker = null;
int totalVaults = 0;
System.out.println("Your Amazon Glacier vaults:");
try {
    while (!listComplete) {
        ListVaultsResponse response = null;
        if (newMarker != null) {
            ListVaultsRequest request = ListVaultsRequest.builder()
                .marker(newMarker)
                .build();

            response = glacier.listVaults(request);
        } else {
            ListVaultsRequest request = ListVaultsRequest.builder()
                .build();
            response = glacier.listVaults(request);
        }

        List<DescribeVaultOutput> vaultList = response.vaultList();
        for (DescribeVaultOutput v : vaultList) {
            totalVaults += 1;
            System.out.println("* " + v.vaultName());
        }

        // Check for further results.
        newMarker = response.marker();
        if (newMarker == null) {
            listComplete = true;
        }
    }

    if (totalVaults == 0) {
        System.out.println("No vaults found.");
    }

} catch (GlacierException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス [ListVaults](#)」の「」を参照してください。AWS SDK for Java 2.x

## UploadArchive

次の例は、UploadArchive を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.UploadArchiveRequest;
import software.amazon.awssdk.services.glacier.model.UploadArchiveResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.io.File;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UploadArchive {

    static final int ONE_MB = 1024 * 1024;

    public static void main(String[] args) {
        final String usage = ""
```



```
Usage:  <strPath> <vaultName>\s

Where:
    strPath - The path to the archive to upload (for example, C:\\AWS
\\test.pdf).
    vaultName - The name of the vault.
    """;

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String strPath = args[0];
String vaultName = args[1];
File myFile = new File(strPath);
Path path = Paths.get(strPath);
GlacierClient glacier = GlacierClient.builder()
    .region(Region.US_EAST_1)
    .build();

String archiveId = uploadContent(glacier, path, vaultName, myFile);
System.out.println("The ID of the archived item is " + archiveId);
glacier.close();
}

public static String uploadContent(GlacierClient glacier, Path path, String
vaultName, File myFile) {
    // Get an SHA-256 tree hash value.
    String checkVal = computeSHA256(myFile);
    try {
        UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
            .vaultName(vaultName)
            .checksum(checkVal)
            .build();

        UploadArchiveResponse res = glacier.uploadArchive(uploadRequest, path);
        return res.archiveId();

    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        return "";
    }

    private static String computeSHA256(File inputFile) {
        try {
            byte[] treeHash = computeSHA256TreeHash(inputFile);
            System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
            return toHex(treeHash);

        } catch (IOException ioe) {
            System.err.format("Exception when reading from file %s: %s", inputFile,
                ioe.getMessage());
            System.exit(-1);

        } catch (NoSuchAlgorithmException nsae) {
            System.err.format("Cannot locate MessageDigest algorithm for SHA-256:
%s", nsae.getMessage());
            System.exit(-1);
        }
        return "";
    }

    public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
        NoSuchAlgorithmException {

        byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
        return computeSHA256TreeHash(chunkSHA256Hashes);
    }

    /**
     * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
     * includes the checksum for the last chunk, even if it's smaller than 1 MB.
     */
    public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
        NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        long numChunks = file.length() / ONE_MB;
        if (file.length() % ONE_MB > 0) {
            numChunks++;
        }

        if (numChunks == 0) {
            return new byte[][] { md.digest() };
        }
    }
}
```

```
    }

    byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
    FileInputStream fileStream = null;

    try {
        fileStream = new FileInputStream(file);
        byte[] buff = new byte[ONE_MB];

        int bytesRead;
        int idx = 0;

        while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
            md.reset();
            md.update(buff, 0, bytesRead);
            chunkSHA256Hashes[idx++] = md.digest();
        }

        return chunkSHA256Hashes;

    } finally {
        if (fileStream != null) {
            try {
                fileStream.close();
            } catch (IOException ioe) {
                System.err.printf("Exception while closing %s.\n %s",
file.getName(),
                                ioe.getMessage());
            }
        }
    }
}

/**
 * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
 * checksums.
 */
public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
    throws NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    byte[][] prevLvlHashes = chunkSHA256Hashes;
    while (prevLvlHashes.length > 1) {
        int len = prevLvlHashes.length / 2;
```

```
        if (prevLvlHashes.length % 2 != 0) {
            len++;
        }

        byte[][] currLvlHashes = new byte[len][];
        int j = 0;
        for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

            // If there are at least two elements remaining.
            if (prevLvlHashes.length - i > 1) {

                // Calculate a digest of the concatenated nodes.
                md.reset();
                md.update(prevLvlHashes[i]);
                md.update(prevLvlHashes[i + 1]);
                currLvlHashes[j] = md.digest();

            } else { // Take care of the remaining odd chunk
                currLvlHashes[j] = prevLvlHashes[i];
            }
        }

        prevLvlHashes = currLvlHashes;
    }

    return prevLvlHashes[0];
}

/**
 * Returns the hexadecimal representation of the input byte array
 */
public static String toHex(byte[] data) {
    StringBuilder sb = new StringBuilder(data.length * 2);
    for (byte datum : data) {
        String hex = Integer.toHexString(datum & 0xFF);

        if (hex.length() == 1) {
            // Append leading zero.
            sb.append("0");
        }
        sb.append(hex);
    }
    return sb.toString().toLowerCase();
}
```

```
}
```

- API の詳細については、「API リファレンス [UploadArchive](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## SageMaker SDK for Java 2.x を使用した の例

次のコード例は、AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています SageMaker。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### 開始方法

こんにちは SageMakerは

次のコード例は、の使用を開始する方法を示しています SageMaker。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 */
```

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

- APIの詳細については、「APIリファレンス[ListNotebookInstances](#)」の「」を参照してください。AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### CreatePipeline

次の例は、CreatePipeline を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Create a pipeline from the example pipeline JSON.
public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
    String functionArn, String pipelineName) {
    System.out.println("Setting up the pipeline.");
    JSONParser parser = new JSONParser();

    // Read JSON and get pipeline definition.
    try (FileReader reader = new FileReader(filePath)) {
        Object obj = parser.parse(reader);
        JSONObject jsonObject = (JSONObject) obj;
        JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
        for (Object stepObj : stepsArray) {
            JSONObject step = (JSONObject) stepObj;
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArn);
            }
        }
        System.out.println(jsonObject);

        // Create the pipeline.
        CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
            .pipelineDescription("Java SDK example pipeline")
            .roleArn(roleArn)
            .pipelineName(pipelineName)
            .pipelineDefinition(jsonObject.toString())
            .build();
    }
}
```

```
sageMakerClient.createPipeline(pipelineRequest);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
} catch (IOException | ParseException e) {
    throw new RuntimeException(e);
}
}
```

- API の詳細については、「API リファレンス [CreatePipeline](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeletePipeline

次の例は、DeletePipeline を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Delete a SageMaker pipeline by name.
public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
    DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
        .pipelineName(pipelineName)
        .build();

    sageMakerClient.deletePipeline(pipelineRequest);
    System.out.println("*** Successfully deleted " + pipelineName);
}
```

- API の詳細については、「API リファレンス [DeletePipeline](#)」の「」を参照してください。  
AWS SDK for Java 2.x



## DescribePipelineExecution

次の例は、DescribePipelineExecution を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// 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);
}
```

- API の詳細については、「API リファレンス [DescribePipelineExecution](#)」の「」を参照してください。 AWS SDK for Java 2.x

## StartPipelineExecution

次の例は、StartPipelineExecution を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

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

- API の詳細については、「API リファレンス[StartPipelineExecution](#)」の「」を参照してください。AWS SDK for Java 2.x

## シナリオ

### 地理空間ジョブとパイプラインの使用を開始する

次のコードサンプルは、以下の操作方法を示しています。

- パイプラインのリソースを設定します。
- 地理空間ジョブを実行するパイプラインを設定します。
- パイプラインの実行を開始します。
- ジョブ実行のステータスをモニタリングします。
- パイプラインの出力を表示します。
- リソースをクリーンアップします。

詳細については、「[Community.aws で AWS SDKs を使用して SageMaker パイプラインを作成および実行する](#)」を参照してください。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public class SagemakerWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String eventSourceMapping = "";

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
            "Usage:\n" +
            "    <sageMakerRoleName> <lambdaRoleName> <functionFileLocation>
<functionName> <queueName> <bucketName> <lnglatData> <spatialPipelinePath>
<pipelineName>\n\n"
            +
            "Where:\n" +
            "    sageMakerRoleName - The name of the Amazon SageMaker role.\n\n"
            +
            "    lambdaRoleName - The name of the AWS Lambda role.\n\n" +
            "    functionFileLocation - The file location where the JAR file
that represents the AWS Lambda function is located.\n\n"
            +
            "    functionName - The name of the AWS Lambda function (for
example, SageMakerExampleFunction).\n\n" +
            "    queueName - The name of the Amazon Simple Queue Service (Amazon
SQS) queue.\n\n" +
            "    bucketName - The name of the Amazon Simple Storage Service
(Amazon S3) bucket.\n\n" +
            "    lnglatData - The file location of the latlongtest.csv file
required for this use case.\n\n" +
            "    spatialPipelinePath - The file location of the
GeoSpatialPipeline.json file required for this use case.\n\n"
            +
            "    pipelineName - The name of the pipeline to create (for example,
sagemaker-sdk-example-pipeline).\n\n";

        if (args.length != 9) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String sageMakerRoleName = args[0];
    String lambdaRoleName = args[1];
    String functionFileLocation = args[2];
    String functionName = args[3];
    String queueName = args[4];
    String bucketName = args[5];
    String lnglatData = args[6];
    String spatialPipelinePath = args[7];
    String pipelineName = args[8];
    String handlerName = "org.example.SageMakerLambdaFunction::handleRequest";

    Region region = Region.US_WEST_2;
    SageMakerClient sageMakerClient = SageMakerClient.builder()
        .region(region)
        .build();

    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    LambdaClient lambdaClient = LambdaClient.builder()
        .region(region)
        .build();

    SqsClient sqsClient = SqsClient.builder()
        .region(region)
        .build();

    S3Client s3Client = S3Client.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon SageMaker pipeline example
scenario.");
    System.out.println(
        "\nThis example workflow will guide you through setting up and
running an" +
        "\nAmazon SageMaker pipeline. The pipeline uses an AWS
Lambda function and an" +
```

```
        "\nAmazon SQS Queue. It runs a vector enrichment reverse
geocode job to" +
        "\nreverse geocode addresses in an input file and store the
results in an export file.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("First, we will set up the roles, functions, and queue
needed by the SageMaker pipeline.");
    String lambdaRoleArn = checkLambdaRole(iam, lambdaRoleName);
    String sageMakerRoleArn = checkSageMakerRole(iam, sageMakerRoleName);

    String functionArn = checkFunction(lambdaClient, functionName,
functionFileLocation, lambdaRoleArn,
        handlerName);
    String queueUrl = checkQueue(sqsClient, lambdaClient, queueName,
functionName);
    System.out.println("The queue URL is " + queueUrl);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Setting up bucket " + bucketName);
    if (!checkBucket(s3Client, bucketName)) {
        setupBucket(s3Client, bucketName);
        System.out.println("Put " + lnglatData + " into " + bucketName);
        putS3Object(s3Client, bucketName, "latlongtest.csv", lnglatData);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Now we can create and run our pipeline.");
    setupPipeline(sageMakerClient, spatialPipelinePath, sageMakerRoleArn,
functionArn, pipelineName);
    String pipelineExecutionARN = executePipeline(sageMakerClient, bucketName,
queueUrl, sageMakerRoleArn,
        pipelineName);
    System.out.println("The pipeline execution ARN value is " +
pipelineExecutionARN);
    waitForPipelineExecution(sageMakerClient, pipelineExecutionARN);
    System.out.println("Getting output results " + bucketName);
    getOutputResults(s3Client, bucketName);
    System.out.println(DASHES);

    System.out.println(DASHES);
```

```
        System.out.println("The pipeline has completed. To view the pipeline and
runs " +
        "in SageMaker Studio, follow these instructions:" +
        "\nhttps://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-
studio.html");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Do you want to delete the AWS resources used in this
Workflow? (y/n)");
        Scanner in = new Scanner(System.in);
        String delResources = in.nextLine();
        if (delResources.compareTo("y") == 0) {
            System.out.println("Lets clean up the AWS resources. Wait 30 seconds");
            TimeUnit.SECONDS.sleep(30);
            deleteEventSourceMapping(lambdaClient);
            deleteSQSQueue(sqsClient, queueName);
            listBucketObjects(s3Client, bucketName);
            deleteBucket(s3Client, bucketName);
            deleteLambdaFunction(lambdaClient, functionName);
            deleteLambdaRole(iam, lambdaRoleName);
            deleteSagemakerRole(iam, sageMakerRoleName);
            deletePipeline(sageMakerClient, pipelineName);
        } else {
            System.out.println("The AWS Resources were not deleted!");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SageMaker pipeline scenario is complete.");
        System.out.println(DASHES);
    }

    private static void readObject(S3Client s3Client, String bucketName, String key)
    {
        System.out.println("Output file contents: \n");
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
        byte[] byteArray = objectBytes.asByteArray();
    }
}
```



```
String text = new String(byteArray, StandardCharsets.UTF_8);
System.out.println("Text output: " + text);
}

// Display some results from the output directory.
public static void getOutputResults(S3Client s3Client, String bucketName) {
    System.out.println("Getting output results {bucketName}.");
    ListObjectsRequest listObjectsRequest = ListObjectsRequest.builder()
        .bucket(bucketName)
        .prefix("outputfiles/")
        .build();

    ListObjectsResponse response = s3Client.listObjects(listObjectsRequest);
    List<S3Object> s3Objects = response.contents();
    for (S3Object object : s3Objects) {
        readObject(s3Client, bucketName, object.key());
    }
}

// Check the status of a pipeline execution.
public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
    throws InterruptedException {
    String status;
    int index = 0;
    do {
        DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
            .pipelineExecutionArn(executionArn)
            .build();

        DescribePipelineExecutionResponse response = sageMakerClient
            .describePipelineExecution(pipelineExecutionRequest);
        status = response.pipelineExecutionStatusAsString();
        System.out.println(index + ". The Status of the pipeline is " + status);
        TimeUnit.SECONDS.sleep(4);
        index++;
    } while ("Executing".equals(status));
    System.out.println("Pipeline finished with status " + status);
}

// Delete a SageMaker pipeline by name.
public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
```

```
DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
    .pipelineName(pipelineName)
    .build();

sageMakerClient.deletePipeline(pipelineRequest);
System.out.println("*** Successfully deleted " + pipelineName);
}

// Create a pipeline from the example pipeline JSON.
public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
    String functionArn, String pipelineName) {
    System.out.println("Setting up the pipeline.");
    JSONParser parser = new JSONParser();

    // Read JSON and get pipeline definition.
    try (FileReader reader = new FileReader(filePath)) {
        Object obj = parser.parse(reader);
        JSONObject jsonObject = (JSONObject) obj;
        JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
        for (Object stepObj : stepsArray) {
            JSONObject step = (JSONObject) stepObj;
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArn);
            }
        }
    }
    System.out.println(jsonObject);

    // Create the pipeline.
    CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
        .pipelineDescription("Java SDK example pipeline")
        .roleArn(roleArn)
        .pipelineName(pipelineName)
        .pipelineDefinition(jsonObject.toString())
        .build();

    sageMakerClient.createPipeline(pipelineRequest);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
} catch (IOException | ParseException e) {
    throw new RuntimeException(e);
}
```

```
}

// Start a pipeline run with job configurations.
public static String executePipeline(SageMakerClient sageMakerClient, String
bucketName, String queueUrl,
    String roleArn, String pipelineName) {
    System.out.println("Starting pipeline execution.");
    String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
    String output = "s3://" + bucketName + "/outputfiles/";
    Gson gson = new GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting().create();

    // Set up all parameters required to start the pipeline.
    List<Parameter> parameters = new ArrayList<>();
    Parameter para1 = Parameter.builder()
        .name("parameter_execution_role")
        .value(roleArn)
        .build();

    Parameter para2 = Parameter.builder()
        .name("parameter_queue_url")
        .value(queueUrl)
        .build();

    String inputJSON = "{\n" +
        "  \"DataSourceConfig\": {\n" +
        "    \"S3Data\": {\n" +
        "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
        "    },\n" +
        "    \"Type\": \"S3_DATA\"\n" +
        "  },\n" +
        "  \"DocumentType\": \"CSV\"\n" +
        "}";

    System.out.println(inputJSON);

    Parameter para3 = Parameter.builder()
        .name("parameter_vej_input_config")
        .value(inputJSON)
        .build();
}
```

```
// Create an ExportVectorEnrichmentJobOutputConfig object.
VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
    .s3Uri(output)
    .build();

ExportVectorEnrichmentJobOutputConfig outputConfig =
ExportVectorEnrichmentJobOutputConfig.builder()
    .s3Data(jobS3Data)
    .build();

String gson4 = gson.toJson(outputConfig);
Parameter para4 = Parameter.builder()
    .name("parameter_vej_export_config")
    .value(gson4)
    .build();

System.out.println("parameter_vej_export_config:" +
gson.toJson(outputConfig));

// Create a VectorEnrichmentJobConfig object.
ReverseGeocodingConfig reverseGeocodingConfig =
ReverseGeocodingConfig.builder()
    .xAttributeName("Longitude")
    .yAttributeName("Latitude")
    .build();

VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
    .reverseGeocodingConfig(reverseGeocodingConfig)
    .build();

String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\": \"Longitude\", \"YAttributeName\": \"Latitude\"}}";
Parameter para5 = Parameter.builder()
    .name("parameter_step_1_vej_config")
    .value(para5JSON)
    .build();

System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
parameters.add(para1);
parameters.add(para2);
parameters.add(para3);
parameters.add(para4);
parameters.add(para5);
```

```
        StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
    .pipelineExecutionDescription("Created using Java SDK")
    .pipelineExecutionDisplayName(pipelineName + "-example-execution")
    .pipelineParameters(parameters)
    .pipelineName(pipelineName)
    .build();

        StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
        return response.pipelineExecutionArn();
    }

    public static void deleteEventSourceMapping(LambdaClient lambdaClient) {
        DeleteEventSourceMappingRequest eventSourceMappingRequest =
DeleteEventSourceMappingRequest.builder()
    .uuid(eventSourceMapping)
    .build();

        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest);
    }

    public static void deleteSagemakerRole(IamClient iam, String roleName) {
        String[] sageMakerRolePolicies = getSageMakerRolePolicies();
        try {
            for (String policy : sageMakerRolePolicies) {
                // First the policy needs to be detached.
                DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
    .policyArn(policy)
    .roleName(roleName)
    .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
    .roleName(roleName)
    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);
        }
    }
}
```

```
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteLambdaRole(IamClient iam, String roleName) {
    String[] lambdaRolePolicies = getLambdaRolePolicies();
    try {
        for (String policy : lambdaRolePolicies) {
            // First the policy needs to be detached.
            DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                .policyArn(policy)
                .roleName(roleName)
                .build();

            iam.detachRolePolicy(rolePolicyRequest);
        }

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Delete the specific AWS Lambda function.
public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("*** " + functionName + " was deleted");
    }
}
```

```
    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Delete the specific S3 bucket.
public static void deleteBucket(S3Client s3Client, String bucketName) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucketName)
        .build();
    s3Client.deleteBucket(deleteBucketRequest);
    System.out.println("*** " + bucketName + " was deleted.");
}

public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.print("\n The name of the key is " + myValue.key());
            deleteBucketObjects(s3, bucketName, myValue.key());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteBucketObjects(S3Client s3, String bucketName, String
objectName) {
    ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
    toDelete.add(ObjectIdentifier.builder()
        .key(objectName)
        .build());
    try {
        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
```

```
        .bucket(bucketName)
        .delete(Delete.builder()
            .objects(toDelete).build())
        .build();

    s3.deleteObjects(dor);
    System.out.println("*** " + bucketName + " objects were deleted.");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

// Delete the specific Amazon SQS queue.
public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
    try {
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
    try {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("x-amz-meta-myVal", "test");
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key("samplefiles/" + objectKey)
            .metadata(metadata)
            .build();
    }
}
```



```
s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void setupBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Set up the SQS queue to use with the pipeline.
public static String setupQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
    String lambdaName) {
    System.out.println("Setting up queue named " + queueName);
    try {
        Map<QueueAttributeName, String> queueAtt = new HashMap<>();
        queueAtt.put(QueueAttributeName.DELAY_SECONDS, "5");
        queueAtt.put(QueueAttributeName.RECEIVE_MESSAGE_WAIT_TIME_SECONDS, "5");
```

```
        queueAtt.put(QueueAttributeName.VISIBILITY_TIMEOUT, "300");
        CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
            .queueName(queueName)
            .attributes(queueAtt)
            .build();

        sqsClient.createQueue(createQueueRequest);
        System.out.println("\nGet queue url");
        GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        TimeUnit.SECONDS.sleep(15);

        connectLambda(sqsClient, lambdaClient, getQueueUrlResponse.queueUrl(),
lambdaName);
        System.out.println("Queue ready with Url " +
getQueueUrlResponse.queueUrl());
        return getQueueUrlResponse.queueUrl();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    return "";
}

// Connect the queue to the Lambda function as an event source.
public static void connectLambda(SqsClient sqsClient, LambdaClient lambdaClient,
String queueUrl,
    String lambdaName) {
    System.out.println("Connecting the Lambda function and queue for the
pipeline.");
    String queueArn = "";

    // Specify the attributes to retrieve.
    List<QueueAttributeName> atts = new ArrayList<>();
    atts.add(QueueAttributeName.QUEUE_ARN);
    GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
        .queueUrl(queueUrl)
        .attributeNames(atts)
        .build();
```

```
    GetQueueAttributesResponse response =
sqscClient.getQueueAttributes(attributesRequest);
    Map<String, String> queueAtts = response.attributesAsStrings();
    for (Map.Entry<String, String> queueAtt : queueAtts.entrySet()) {
        System.out.println("Key = " + queueAtt.getKey() + ", Value = " +
queueAtt.getValue());
        queueArn = queueAtt.getValue();
    }

    CreateEventSourceMappingRequest eventSourceMappingRequest =
CreateEventSourceMappingRequest.builder()
        .eventSourceArn(queueArn)
        .functionName(lambdaName)
        .build();

    CreateEventSourceMappingResponse response1 =
lambdaClient.createEventSourceMapping(eventSourceMappingRequest);
    eventSourceMapping = response1.uuid();
    System.out.println("The mapping between the event source and Lambda function
was successful");
}

// Create an AWS Lambda function.
public static String createLambdaFunction(LambdaClient awsLambda, String
functionName, String filePath, String role,
String handler) {
    try {
        LambdaWaiter waiter = awsLambda.waiter();
        InputStream is = new FileInputStream(filePath);
        SdkBytes fileToUpload = SdkBytes.fromInputStream(is);
        FunctionCode code = FunctionCode.builder()
            .zipFile(fileToUpload)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .functionName(functionName)
            .description("SageMaker example function.")
            .code(code)
            .handler(handler)
            .runtime(Runtime.JAVA11)
            .timeout(200)
            .memorySize(1024)
            .role(role)
    }
```

```

        .build());

        // Create a Lambda function using a waiter.
        CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
        GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();
        WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The function ARN is " +
functionResponse.functionArn());
        return functionResponse.functionArn();

    } catch (LambdaException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static String createSageMakerRole(IamClient iam, String roleName) {
    String[] sageMakerRolePolicies = getSageMakerRolePolicies();
    System.out.println("Creating a role to use with SageMaker.");
    String assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\", " +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\", " +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\", " +
        "\"sagemaker-geospatial.amazonaws.com\", " +
        "\"lambda.amazonaws.com\", " +
        "\"s3.amazonaws.com\" " +
        "]" +
        "}, " +
        "\"Action\": \"sts:AssumeRole\" " +
        "}] " +
        "}";

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(roleName)

```

```

        .assumeRolePolicyDocument(assumeRolePolicy)
        .description("Created using the AWS SDK for Java")
        .build();

    CreateRoleResponse roleResult = iam.createRole(request);

    // Attach the policies to the role.
    for (String policy : sageMakerRolePolicies) {
        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policy)
            .build();

        iam.attachRolePolicy(attachRequest);
    }

    // Allow time for the role to be ready.
    TimeUnit.SECONDS.sleep(15);
    System.out.println("Role ready with ARN " + roleResult.role().arn());
    return roleResult.role().arn();

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
} catch (InterruptedException e) {
    throw new RuntimeException(e);
}
return "";
}

private static String createLambdaRole(IamClient iam, String roleName) {
    String[] lambdaRolePolicies = getLambdaRolePolicies();
    String assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\", " +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\", " +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\", " +
        "\"sagemaker-geospatial.amazonaws.com\", " +
        "\"lambda.amazonaws.com\", " +
        "\"s3.amazonaws.com\" " +
        "]" +
    "}" +

```

```
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]" +
        "}";

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(roleName)
            .assumeRolePolicyDocument(assumeRolePolicy)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse roleResult = iam.createRole(request);

        // Attach the policies to the role.
        for (String policy : lambdaRolePolicies) {
            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policy)
                .build();

            iam.attachRolePolicy(attachRequest);
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15);
        System.out.println("Role ready with ARN " + roleResult.role().arn());
        return roleResult.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    return "";
}

public static String checkFunction(LambdaClient lambdaClient, String
functionName, String filePath, String role,
    String handler) {
    System.out.println("Create an AWS Lambda function used in this workflow.");
    String functionArn;
```

```
    try {
        // Does this function already exist.
        GetFunctionRequest functionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();

        GetFunctionResponse response =
lambdaClient.getFunction(functionRequest);
        functionArn = response.configuration().functionArn();

    } catch (LambdaException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        functionArn = createLambdaFunction(lambdaClient, functionName, filePath,
role, handler);
    }
    return functionArn;
}

// Check to see if the specific S3 bucket exists. If the S3 bucket exists, this
// method returns true.
public static boolean checkBucket(S3Client s3, String bucketName) {
    try {
        HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3.headBucket(headBucketRequest);
        System.out.println(bucketName + " exists");
        return true;

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    return false;
}

// Checks to see if the Amazon SQS queue exists. If not, this method creates a
// new queue
// and returns the ARN value.
public static String checkQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
    String lambdaName) {
    System.out.println("Creating a queue for this use case.");
    String queueUrl;
```

```
    try {
        GetQueueUrlRequest request = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        GetQueueUrlResponse response = sqsClient.getQueueUrl(request);
        queueUrl = response.queueUrl();
        System.out.println(queueUrl);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        queueUrl = setupQueue(sqsClient, lambdaClient, queueName, lambdaName);
    }
    return queueUrl;
}

// Checks to see if the Lambda role exists. If not, this method creates it.
public static String checkLambdaRole(IamClient iam, String roleName) {
    System.out.println("Creating a role to for AWS Lambda to use.");
    String roleArn;
    try {
        GetRoleRequest roleRequest = GetRoleRequest.builder()
            .roleName(roleName)
            .build();

        GetRoleResponse response = iam.getRole(roleRequest);
        roleArn = response.role().arn();
        System.out.println(roleArn);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        roleArn = createLambdaRole(iam, roleName);
    }
    return roleArn;
}

// Checks to see if the SageMaker role exists. If not, this method creates it.
public static String checkSageMakerRole(IamClient iam, String roleName) {
    System.out.println("Creating a role to for AWS SageMaker to use.");
    String roleArn;
    try {
        GetRoleRequest roleRequest = GetRoleRequest.builder()
            .roleName(roleName)
            .build();
```



```
        GetRoleResponse response = iam.getRole(roleRequest);
        roleArn = response.role().arn();
        System.out.println(roleArn);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        roleArn = createSageMakerRole(iam, roleName);
    }
    return roleArn;
}

private static String[] getSageMakerRolePolicies() {
    String[] sageMakerRolePolicies = new String[3];
    sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
    sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/" +
    "AmazonSageMakerGeospatialFullAccess";
    sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
    return sageMakerRolePolicies;
}

private static String[] getLambdaRolePolicies() {
    String[] lambdaRolePolicies = new String[5];
    lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
    lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
    lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
    "AmazonSageMakerGeospatialFullAccess";
    lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/"
        + "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy";
    lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
    "AWSLambdaSQSQueueExecutionRole";
    return lambdaRolePolicies;
}
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [CreatePipeline](#)
  - [DeletePipeline](#)
  - [DescribePipelineExecution](#)

- [StartPipelineExecution](#)
- [UpdatePipeline](#)

## SDK for Java 2.x を使用した Secrets Manager の例

次のコード例は、Secrets Manager AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- [アクション](#)

### アクション

#### GetSecretValue

次の例は、GetSecretValue を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
```

```
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * We recommend that you cache your secret values by using client-side caching.
 *
 * Caching secrets improves speed and reduces your costs. For more information,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/retrieving-secrets.html
 */
public class GetSecretValue {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <secretName>\s

                Where:
                secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String secretName = args[0];
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
                .region(region)
                .build();

        getValue(secretsClient, secretName);
        secretsClient.close();
    }
}
```

```
public static void getValue(SecretsManagerClient secretsClient, String
secretName) {
    try {
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
secretsClient.getSecretValue(valueRequest);
        String secret = valueResponse.secretString();
        System.out.println(secret);

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [GetSecretValue](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## SDK for Java 2.x を使用する Amazon SES の例

次のコード例は、Amazon SES AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### トピック

- [アクション](#)

## アクション

### ListIdentities

次の例は、ListIdentities を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.ListIdentitiesResponse;
import software.amazon.awssdk.services.ses.model.SesException;
import java.io.IOException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentities {

    public static void main(String[] args) throws IOException {
        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
            .region(region)
            .build();

        listSESIIdentities(client);
    }

    public static void listSESIIdentities(SesClient client) {
        try {
            ListIdentitiesResponse identitiesResponse = client.listIdentities();
        }
    }
}
```

```
        List<String> identities = identitiesResponse.identities();
        for (String identity : identities) {
            System.out.println("The identity is " + identity);
        }

    } catch (SesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [ListIdentities](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListTemplates

次の例は、ListTemplates を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesRequest;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesResponse;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;

public class ListTemplates {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
            .region(region)
            .build();
    }
}
```

```
        listAllTemplates(sesv2Client);
    }

    public static void listAllTemplates(SesV2Client sesv2Client) {
        try {
            ListEmailTemplatesRequest templatesRequest =
                ListEmailTemplatesRequest.builder()
                    .pageSize(1)
                    .build();

            ListEmailTemplatesResponse response =
                sesv2Client.listEmailTemplates(templatesRequest);
            response.templatesMetadata()
                .forEach(template -> System.out.println("Template name: " +
                    template.templateName()));

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [List Templates](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## SendEmail

次の例は、SendEmail を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.Content;
import software.amazon.awssdk.services.ses.model.Destination;
import software.amazon.awssdk.services.ses.model.Message;
import software.amazon.awssdk.services.ses.model.Body;
import software.amazon.awssdk.services.ses.model.SendEmailRequest;
import software.amazon.awssdk.services.ses.model.SesException;

import javax.mail.MessagingException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageEmailRequest {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <sender> <recipient> <subject>\s

            Where:
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
                subject - The subject line.\s

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesClient client = SesClient.builder()
            .region(region)
            .build();
```



```
// The HTML body of the email.
String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
    + "<p> See the list of customers.</p>" + "</body>" + "</html>";

try {
    send(client, sender, recipient, subject, bodyHTML);
    client.close();
    System.out.println("Done");
} catch (MessagingException e) {
    e.printStackTrace();
}
}

public static void send(SesClient client,
    String sender,
    String recipient,
    String subject,
    String bodyHTML) throws MessagingException {

    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    Content content = Content.builder()
        .data(bodyHTML)
        .build();

    Content sub = Content.builder()
        .data(subject)
        .build();

    Body body = Body.builder()
        .html(content)
        .build();

    Message msg = Message.builder()
        .subject(sub)
        .body(body)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
```

```
        .message(msg)
        .source(sender)
        .build();

    try {
        System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");
        client.sendEmail(emailRequest);

    } catch (SesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.mail.internet.MimeBodyPart;
import javax.mail.util.ByteArrayDataSource;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.file.Files;
import java.util.Properties;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.ses.model.SendRawEmailRequest;
import software.amazon.awssdk.services.ses.model.RawMessage;
import software.amazon.awssdk.services.ses.model.SesException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
*  
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
*/
```

```
public class SendMessageAttachment {  
    public static void main(String[] args) throws IOException {  
        final String usage = ""  
  
            Usage:  
            <sender> <recipient> <subject> <fileLocation>\s  
  
            Where:  
            sender - An email address that represents the sender.\s  
            recipient - An email address that represents the recipient.\s  
            subject - The subject line.\s  
            fileLocation - The location of a Microsoft Excel file to use as  
an attachment (C:/AWS/customers.xls).\s  
            "";  
  
        if (args.length != 4) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        String sender = args[0];  
        String recipient = args[1];  
        String subject = args[2];  
        String fileLocation = args[3];  
  
        // The email body for recipients with non-HTML email clients.  
        String bodyText = "Hello,\r\n" + "Please see the attached file for a list "  
            + "of customers to contact.";  
  
        // The HTML body of the email.  
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"  
            + "<p>Please see the attached file for a " + "list of customers to  
contact.</p>" + "</body>"  
            + "</html>";  
  
        Region region = Region.US_WEST_2;  
        SesClient client = SesClient.builder()  
            .region(region)  
            .build();
```

```
        try {
            sendemailAttachment(client, sender, recipient, subject, bodyText,
bodyHTML, fileLocation);
            client.close();
            System.out.println("Done");
        } catch (IOException | MessagingException e) {
            e.printStackTrace();
        }
    }
}

public static void sendemailAttachment(SesClient client,
    String sender,
    String recipient,
    String subject,
    String bodyText,
    String bodyHTML,
    String fileLocation) throws AddressException, MessagingException,
IOException {

    java.io.File theFile = new java.io.File(fileLocation);
    byte[] fileContent = Files.readAllBytes(theFile.toPath());

    Session session = Session.getDefaultInstance(new Properties());

    // Create a new MimeMessage object.
    MimeMessage message = new MimeMessage(session);

    // Add subject, from and to lines.
    message.setSubject(subject, "UTF-8");
    message.setFrom(new InternetAddress(sender));
    message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(recipient));

    // Create a multipart/alternative child container.
    MimeMultipart msgBody = new MimeMultipart("alternative");

    // Create a wrapper for the HTML and text parts.
    MimeBodyPart wrap = new MimeBodyPart();

    // Define the text part.
    MimeBodyPart textPart = new MimeBodyPart();
    textPart.setContent(bodyText, "text/plain; charset=UTF-8");
```

```
// Define the HTML part.
MimeBodyPart htmlPart = new MimeBodyPart();
htmlPart.setContent(bodyHTML, "text/html; charset=UTF-8");

// Add the text and HTML parts to the child container.
msgBody.addBodyPart(textPart);
msgBody.addBodyPart(htmlPart);

// Add the child container to the wrapper object.
wrap.setContent(msgBody);

// Create a multipart/mixed parent container.
MimeMultipart msg = new MimeMultipart("mixed");

// Add the parent container to the message.
message.setContent(msg);
msg.addBodyPart(wrap);

// Define the attachment.
MimeBodyPart att = new MimeBodyPart();
DataSource fds = new ByteArrayDataSource(fileContent,
    "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet");
att.setDataHandler(new DataHandler(fds));

String reportName = "WorkReport.xls";
att.setFileName(reportName);

// Add the attachment to the message.
msg.addBodyPart(att);

try {
    System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");

    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    message.writeTo(outputStream);

    ByteBuffer buf = ByteBuffer.wrap(outputStream.toByteArray());

    byte[] arr = new byte[buf.remaining()];
    buf.get(arr);

    SdkBytes data = SdkBytes.fromByteArray(arr);
```

```
RawMessage rawMessage = RawMessage.builder()
    .data(data)
    .build();

SendRawEmailRequest rawEmailRequest = SendRawEmailRequest.builder()
    .rawMessage(rawMessage)
    .build();

client.sendRawEmail(rawEmailRequest);

} catch (SesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.println("Email sent using SesClient with attachment");
}
```

- API の詳細については、「API リファレンス [SendEmail](#)」の「」を参照してください。AWS SDK for Java 2.x

## SendTemplatedEmail

次の例は、SendTemplatedEmail を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.Template;
```

```
/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Also, make sure that you create a template. See the following documentation
 * topic:
 *
 * https://docs.aws.amazon.com/ses/latest/dg/send-personalized-email-api.html
 */

public class SendEmailTemplate {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <template> <sender> <recipient>\s

            Where:
                template - The name of the email template.
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String templateName = args[0];
        String sender = args[1];
        String recipient = args[2];
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
            .region(region)
            .build();

        send(sesv2Client, sender, recipient, templateName);
    }
}
```

```
public static void send(SesV2Client client, String sender, String recipient,
String templateName) {
    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    /*
     * Specify both name and favorite animal (favoriteanimal) in your code when
     * defining the Template object.
     * If you don't specify all the variables in the template, Amazon SES
doesn't
     * send the email.
     */
    Template myTemplate = Template.builder()
        .templateName(templateName)
        .templateData("{\n" +
            "  \"name\": \"Jason\"\n," +
            "  \"favoriteanimal\": \"Cat\"\n" +
            "}")
        .build();

    EmailContent emailContent = EmailContent.builder()
        .template(myTemplate)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
        .content(emailContent)
        .fromEmailAddress(sender)
        .build();

    try {
        System.out.println("Attempting to send an email based on a template
using the AWS SDK for Java (v2)...");
        client.sendEmail(emailRequest);
        System.out.println("email based on a template was sent");
    } catch (SesV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



- API の詳細については、「API リファレンス [SendTemplatedEmail](#)」の「」を参照してください。AWS SDK for Java 2.x

## SDK for Java 2.x を使用する Amazon SES API v2 の例

次のコード例は、Amazon SES API v2 AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。Amazon SES

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### CreateContact

次の例は、CreateContact を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
try {
```

```
// Create a new contact with the provided email address in the
CreateContactRequest contactRequest = CreateContactRequest.builder()
    .contactListName(CONTACT_LIST_NAME)
    .emailAddress(emailAddress)
    .build();

sesClient.createContact(contactRequest);
contacts.add(emailAddress);

System.out.println("Contact created: " + emailAddress);

// Send a welcome email to the new contact
String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
    .fromEmailAddress(this.verifiedEmail)
    .destination(Destination.builder().toAddresses(emailAddress).build())
    .content(EmailContent.builder()
        .simple(
            Message.builder()
                .subject(Content.builder().data("Welcome to the Weekly
Coupons Newsletter").build())
                .body(Body.builder()
                    .text(Content.builder().data(welcomeText).build())
                    .html(Content.builder().data(welcomeHtml).build())
                    .build())
                .build()
            )
        .build()
    )
    .build();
SendEmailResponse welcomeEmailResponse =
sesClient.sendEmail(welcomeEmailRequest);
System.out.println("Welcome email sent: " +
welcomeEmailResponse.messageId());
} catch (AlreadyExistsException e) {
    // If the contact already exists, skip this step for that contact and
    proceed
    // with the next contact
    System.out.println("Contact already exists, skipping creation...");
} catch (Exception e) {
    System.err.println("Error occurred while processing email address " +
emailAddress + ": " + e.getMessage());
```

```
        throw e;
    }
}
```

- API の詳細については、「API リファレンス [CreateContact](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateContactList

次の例は、CreateContactList を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
try {
    // 2. Create a contact list
    String contactListName = CONTACT_LIST_NAME;
    CreateContactListRequest createContactListRequest =
CreateContactListRequest.builder()
        .contactListName(contactListName)
        .build();
    sesClient.createContactList(createContactListRequest);
    System.out.println("Contact list created: " + contactListName);
} catch (AlreadyExistsException e) {
    System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
} catch (LimitExceededException e) {
    System.err.println("Limit for contact lists has been exceeded.");
    throw e;
} catch (SesV2Exception e) {
    System.err.println("Error creating contact list: " + e.getMessage());
    throw e;
}
```

- API の詳細については、「API リファレンス [CreateContactList](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateEmailIdentity

次の例は、CreateEmailIdentity を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
try {
    CreateEmailIdentityRequest createEmailIdentityRequest =
CreateEmailIdentityRequest.builder()
        .emailIdentity(verifiedEmail)
        .build();
    sesClient.createEmailIdentity(createEmailIdentityRequest);
    System.out.println("Email identity created: " + verifiedEmail);
} catch (AlreadyExistsException e) {
    System.out.println("Email identity already exists, skipping creation: " +
verifiedEmail);
} catch (NotFoundException e) {
    System.err.println("The provided email address is not verified: " +
verifiedEmail);
    throw e;
} catch (LimitExceededException e) {
    System.err
        .println("You have reached the limit for email identities. Please remove
some identities and try again.");
    throw e;
} catch (SesV2Exception e) {
    System.err.println("Error creating email identity: " + e.getMessage());
    throw e;
}
```

- API の詳細については、「API リファレンス [CreateEmailIdentity](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateEmailTemplate

次の例は、CreateEmailTemplate を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
try {
    // Create an email template named "weekly-coupons"
    String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
    String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");

    CreateEmailTemplateRequest templateRequest =
CreateEmailTemplateRequest.builder()
        .templateName(TEMPLATE_NAME)
        .templateContent(EmailTemplateContent.builder()
            .subject("Weekly Coupons Newsletter")
            .html(newsletterHtml)
            .text(newsletterText)
            .build())
        .build();

    sesClient.createEmailTemplate(templateRequest);

    System.out.println("Email template created: " + TEMPLATE_NAME);
} catch (AlreadyExistsException e) {
    // If the template already exists, skip this step and proceed with the next
    // operation
    System.out.println("Email template already exists, skipping creation...");
} catch (LimitExceededException e) {
    // If the limit for email templates is exceeded, fail the workflow and inform
    // the user
```

```
System.err.println("You have reached the limit for email templates. Please
remove some templates and try again.");
    throw e;
} catch (Exception e) {
    System.err.println("Error occurred while creating email template: " +
e.getMessage());
    throw e;
}
```

- APIの詳細については、「APIリファレンス[CreateEmailTemplate](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteContactList

次の例は、DeleteContactList を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
try {
    // Delete the contact list
    DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
        .contactListName(CONTACT_LIST_NAME)
        .build();

    sesClient.deleteContactList(deleteContactListRequest);

    System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
} catch (NotFoundException e) {
    // If the contact list does not exist, log the error and proceed
    System.out.println("Contact list not found. Skipping deletion...");
} catch (Exception e) {
    System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
}
```

```
e.printStackTrace();
}
```

- API の詳細については、「API リファレンス [DeleteContactList](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteEmailIdentity

次の例は、DeleteEmailIdentity を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
try {
    // Delete the email identity
    DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
    .emailIdentity(this.verifiedEmail)
    .build();

    sesClient.deleteEmailIdentity(deleteIdentityRequest);

    System.out.println("Email identity deleted: " + this.verifiedEmail);
} catch (NotFoundException e) {
    // If the email identity does not exist, log the error and proceed
    System.out.println("Email identity not found. Skipping deletion...");
} catch (Exception e) {
    System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
    e.printStackTrace();
}
} else {
    System.out.println("Skipping email identity deletion.");
}
```

- API の詳細については、「API リファレンス [DeleteEmailIdentity](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteEmailTemplate

次の例は、DeleteEmailTemplate を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
try {
    // Delete the template
    DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
    .templateName(TEMPLATE_NAME)
    .build();

    sesClient.deleteEmailTemplate(deleteTemplateRequest);

    System.out.println("Email template deleted: " + TEMPLATE_NAME);
} catch (NotFoundException e) {
    // If the email template does not exist, log the error and proceed
    System.out.println("Email template not found. Skipping deletion...");
} catch (Exception e) {
    System.err.println("Error occurred while deleting the email template: " +
e.getMessage());
    e.printStackTrace();
}
```

- API の詳細については、「API リファレンス [DeleteEmailTemplate](#)」の「」を参照してください。AWS SDK for Java 2.x



## ListContacts

次の例は、ListContacts を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
ListContactsRequest contactListRequest = ListContactsRequest.builder()
    .contactListName(CONTACT_LIST_NAME)
    .build();

List<String> contactEmails;
try {
    ListContactsResponse contactListResponse =
sesClient.listContacts(contactListRequest);

    contactEmails = contactListResponse.contacts().stream()
        .map(Contact::emailAddress)
        .toList();
} catch (Exception e) {
    // TODO: Remove when listContacts's GET body issue is resolved.
    contactEmails = this.contacts;
}
```

- API の詳細については、「API リファレンス [ListContacts](#)」の「」を参照してください。 AWS SDK for Java 2.x

## SendEmail

次の例は、SendEmail を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

メッセージを送信します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Body;
import software.amazon.awssdk.services.sesv2.model.Content;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.Message;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SendEmail {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <sender> <recipient> <subject>\s

            Where:
                sender - An email address that represents the
sender.\s

                recipient - An email address that represents the
recipient.\s

                subject - The subject line.\s

            """;
```

```
        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
            .region(region)
            .build();

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" +
"<h1>Hello!</h1>"
            + "<p> See the list of customers.</p>" + "</body>" +
"</html>";

        send(sesv2Client, sender, recipient, subject, bodyHTML);
    }

    public static void send(SesV2Client client,
        String sender,
        String recipient,
        String subject,
        String bodyHTML) {

        Destination destination = Destination.builder()
            .toAddresses(recipient)
            .build();

        Content content = Content.builder()
            .data(bodyHTML)
            .build();

        Content sub = Content.builder()
            .data(subject)
            .build();

        Body body = Body.builder()
            .html(content)
```

```
        .build();

    Message msg = Message.builder()
        .subject(sub)
        .body(body)
        .build();

    EmailContent emailContent = EmailContent.builder()
        .simple(msg)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
        .content(emailContent)
        .fromEmailAddress(sender)
        .build();

    try {
        System.out.println("Attempting to send an email through
Amazon SES "
            + "using the AWS SDK for Java...");
        client.sendEmail(emailRequest);
        System.out.println("email was sent");

    } catch (SesV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

テンプレートを使用してメッセージを送信します。

```
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());
}
```

- API の詳細については、「API リファレンス [SendEmail](#)」の「」を参照してください。AWS SDK for Java 2.x

## シナリオ

### ニュースレターワークフロー

次のコード例は、Amazon SES API v2 ニュースレターワークフローの方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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())
```

```
        .build()
        .fromEmailAddress(this.verifiedEmail)
        .listManagementOptions(ListManagementOptions.builder()
            .contactListName(CONTACT_LIST_NAME)
            .build())
        .build();
    SendEmailResponse newsletterResponse =
sesClient.sendEmail(newsletterRequest);
    System.out.println("Newsletter sent to " + emailAddress + ": " +
newsletterResponse.messageId());
    }

    try {
        CreateEmailIdentityRequest createEmailIdentityRequest =
CreateEmailIdentityRequest.builder()
            .emailIdentity(verifiedEmail)
            .build();
        sesClient.createEmailIdentity(createEmailIdentityRequest);
        System.out.println("Email identity created: " + verifiedEmail);
    } catch (AlreadyExistsException e) {
        System.out.println("Email identity already exists, skipping creation: " +
verifiedEmail);
    } catch (NotFoundException e) {
        System.err.println("The provided email address is not verified: " +
verifiedEmail);
        throw e;
    } catch (LimitExceededException e) {
        System.err
            .println("You have reached the limit for email identities. Please remove
some identities and try again.");
        throw e;
    } catch (SesV2Exception e) {
        System.err.println("Error creating email identity: " + e.getMessage());
        throw e;
    }

    try {
        // Create an email template named "weekly-coupons"
        String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
        String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");
```



```
        CreateEmailTemplateRequest templateRequest =
CreateEmailTemplateRequest.builder()
    .templateName(TEMPLATE_NAME)
    .templateContent(EmailTemplateContent.builder()
        .subject("Weekly Coupons Newsletter")
        .html(newsletterHtml)
        .text(newsletterText)
        .build())
    .build();

        sesClient.createEmailTemplate(templateRequest);

        System.out.println("Email template created: " + TEMPLATE_NAME);
    } catch (AlreadyExistsException e) {
        // If the template already exists, skip this step and proceed with the next
        // operation
        System.out.println("Email template already exists, skipping creation...");
    } catch (LimitExceededException e) {
        // If the limit for email templates is exceeded, fail the workflow and inform
        // the user
        System.err.println("You have reached the limit for email templates. Please
remove some templates and try again.");
        throw e;
    } catch (Exception e) {
        System.err.println("Error occurred while creating email template: " +
e.getMessage());
        throw e;
    }
}

        try {
            // Delete the contact list
            DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
                .contactListName(CONTACT_LIST_NAME)
                .build();

            sesClient.deleteContactList(deleteContactListRequest);

            System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
        } catch (NotFoundException e) {
            // If the contact list does not exist, log the error and proceed
            System.out.println("Contact list not found. Skipping deletion...");
        } catch (Exception e) {
```

```
        System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
        e.printStackTrace();
    }

    try {
        // Delete the email identity
        DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
        .emailIdentity(this.verifiedEmail)
        .build();

        sesClient.deleteEmailIdentity(deleteIdentityRequest);

        System.out.println("Email identity deleted: " + this.verifiedEmail);
    } catch (NotFoundException e) {
        // If the email identity does not exist, log the error and proceed
        System.out.println("Email identity not found. Skipping deletion...");
    } catch (Exception e) {
        System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
        e.printStackTrace();
    }
} else {
    System.out.println("Skipping email identity deletion.");
}

    try {
        // Delete the template
        DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
        .templateName(TEMPLATE_NAME)
        .build();

        sesClient.deleteEmailTemplate(deleteTemplateRequest);

        System.out.println("Email template deleted: " + TEMPLATE_NAME);
    } catch (NotFoundException e) {
        // If the email template does not exist, log the error and proceed
        System.out.println("Email template not found. Skipping deletion...");
    } catch (Exception e) {
        System.err.println("Error occurred while deleting the email template: " +
e.getMessage());
        e.printStackTrace();
    }
}
```

```
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [CreateContact](#)
  - [CreateContactList](#)
  - [CreateEmailIdentity](#)
  - [CreateEmailTemplate](#)
  - [DeleteContactList](#)
  - [DeleteEmailIdentity](#)
  - [DeleteEmailTemplate](#)
  - [ListContacts](#)
  - [SendEmail](#) シンプル
  - [SendEmail](#) テンプレート

## SDK for Java 2.x を使用する Amazon SNS の例

次のコード例は、Amazon SNS AWS SDK for Java 2.x を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### 開始方法

#### Hello Amazon SNS

以下のコード例は、Amazon SNS の使用を開始する方法を示しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                .flatMap(r -> r.topics().stream())
                .forEach(content -> System.out.println(" Topic ARN: " +
content.topicArn()));

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [ListTopics](#)」の「」を参照してください。AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)
- [サーバーレスサンプル](#)

## アクション

### CheckIfPhoneNumberIsOptedOut

次の例は、CheckIfPhoneNumberIsOptedOut を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
    software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
    software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CheckOptOut {
```

```
public static void main(String[] args) {

    final String usage = ""

        Usage:    <phoneNumber>

        Where:
            phoneNumber - The mobile phone number to look up (for example,
+1XXX5550100).

        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String phoneNumber = args[0];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    checkPhone(snsClient, phoneNumber);
    snsClient.close();
}

public static void checkPhone(SnsClient snsClient, String phoneNumber) {
    try {
        CheckIfPhoneNumberIsOptedOutRequest request =
CheckIfPhoneNumberIsOptedOutRequest.builder()
            .phoneNumber(phoneNumber)
            .build();

        CheckIfPhoneNumberIsOptedOutResponse result =
snsClient.checkIfPhoneNumberIsOptedOut(request);
        System.out.println(
            result.isOptedOut() + "Phone Number " + phoneNumber + " has
Opted Out of receiving sns messages." +
                "\n\nStatus was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
}
```

- API の詳細については、「API リファレンス [CheckIfPhoneNumberIsOptedOut](#)」の「」を参照してください。AWS SDK for Java 2.x

## ConfirmSubscription

次の例は、ConfirmSubscription を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sns.SnsClient;  
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionRequest;  
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionResponse;  
import software.amazon.awssdk.services.sns.model.SnsException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class ConfirmSubscription {  
    public static void main(String[] args) {  
        final String usage = ""  
  
                Usage:    <subscriptionToken> <topicArn>  
  
                Where:
```

```
        subscriptionToken - A short-lived token sent to an endpoint
during the Subscribe action.
        topicArn - The ARN of the topic.\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String subscriptionToken = args[0];
    String topicArn = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    confirmSub(snsClient, subscriptionToken, topicArn);
    snsClient.close();
}

public static void confirmSub(SnsClient snsClient, String subscriptionToken,
String topicArn) {
    try {
        ConfirmSubscriptionRequest request =
ConfirmSubscriptionRequest.builder()
            .token(subscriptionToken)
            .topicArn(topicArn)
            .build();

        ConfirmSubscriptionResponse result =
snsClient.confirmSubscription(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n"
            + result.subscriptionArn());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```



- API の詳細については、「API リファレンス [ConfirmSubscription](#)」の「」を参照してください。AWS SDK for Java 2.x

## CreateTopic

次の例は、CreateTopic を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <topicName>

                Where:
                    topicName - The name of the topic to create (for example,
mytopic).

                """;

        if (args.length != 1) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String topicName = args[0];
    System.out.println("Creating a topic with name: " + topicName);
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    String arnVal = createSNSTopic(snsClient, topicName);
    System.out.println("The topic ARN is" + arnVal);
    snsClient.close();
}

public static String createSNSTopic(SnsClient snsClient, String topicName) {
    CreateTopicResponse result;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- APIの詳細については、「API リファレンス [CreateTopic](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteTopic

次の例は、DeleteTopic を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn>

            Where:
                topicArn - The ARN of the topic to delete.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
```

```
        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [DeleteTopic](#)」の「」を参照してください。AWS SDK for Java 2.x

## GetSMSAttributes

次の例は、GetSMSAttributes を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
```

```
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetSMSAttributes {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn>

            Where:
                topicArn - The ARN of the topic from which to retrieve
attributes.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        getSNSAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSAttributes(SnsClient snsClient, String topicArn) {
        try {
            GetSubscriptionAttributesRequest request =
GetSubscriptionAttributesRequest.builder()
                .subscriptionArn(topicArn)
                .build();
```

```
        // Get the Subscription attributes
        GetSubscriptionAttributesResponse res =
snsClient.getSubscriptionAttributes(request);
        Map<String, String> map = res.attributes();

        // Iterate through the map
        Iterator iter = map.entrySet().iterator();
        while (iter.hasNext()) {
            Map.Entry entry = (Map.Entry) iter.next();
            System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
entry.getValue());
        }

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("\n\nStatus was good");
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[GetSMSAttributes](#)」を参照してください。

## GetTopicAttributes

次の例は、GetTopicAttributes を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
```

```
import software.amazon.awssdk.services.sns.model.GetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetTopicAttributes {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn>

            Where:
                topicArn - The ARN of the topic to look up.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        System.out.println("Getting attributes for a topic with name: " + topicArn);
        getSNSTopicAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSTopicAttributes(SnsClient snsClient, String topicArn) {
        try {
            GetTopicAttributesRequest request = GetTopicAttributesRequest.builder()
                .topicArn(topicArn)
                .build();
        }
    }
}
```

```
        GetTopicAttributesResponse result =
snsClient.getTopicAttributes(request);
        System.out.println("\n\nStatus is " +
result.sdkHttpResponse().statusCode() + "\n\nAttributes: \n\n"
            + result.attributes());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [GetTopicAttributes](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListPhoneNumbersOptedOut

次の例は、ListPhoneNumbersOptedOut を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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();

        listOptOut(snsClient);
        snsClient.close();
    }

    public static void listOptOut(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
                ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
                snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " + result.sdkHttpResponse().statusCode()
                + "\n\nPhone Numbers: \n\n"
                + result.phoneNumbers());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- APIの詳細については、「APIリファレンス[ListPhoneNumbersOptedOut](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListSubscriptions

次の例は、ListSubscriptions を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsRequest;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListSubscriptions {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listSNSSubscriptions(snsClient);
        snsClient.close();
    }

    public static void listSNSSubscriptions(SnsClient snsClient) {
        try {
            ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
                .build();

            ListSubscriptionsResponse result = snsClient.listSubscriptions(request);
            System.out.println(result.subscriptions());
        } catch (SnsException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [ListSubscriptions](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListTopics

次の例は、ListTopics を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTopics {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
}
```

```
        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
                .build();

            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println(
                "Status was " + result.sdkHttpResponse().statusCode() + "\n\n"
                + result.topics());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [ListTopics](#)」の「」を参照してください。AWS SDK for Java 2.x

## Publish

次の例は、Publish を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
```

```
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <message> <topicArn>

            Where:
                message - The message text to send.
                topicArn - The ARN of the topic to publish.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        pubTopic(snsClient, message, topicArn);
        snsClient.close();
    }

    public static void pubTopic(SnsClient snsClient, String message, String
topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .topicArn(topicArn)
                .build();

```

```
        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[Publish](#)」を参照してください。

## SetSMSAttributes

次の例は、SetSMSAttributes を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String, String>
attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                .attributes(attributes)
                .build();

            SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ". Status
was "
                + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- APIの詳細については、AWS SDK for Java 2.x API リファレンスの「[SetSMSAttributes](#)」を参照してください。

## SetSubscriptionAttributes

次の例は、SetSubscriptionAttributes を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <subscriptionArn>

            Where:
                subscriptionArn - The ARN of a subscription.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
```



```
        usePolicy(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
        try {
            SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
            // Add a filter policy attribute with a single value
            fp.addAttribute("store", "example_corp");
            fp.addAttribute("event", "order_placed");

            // Add a prefix attribute
            fp.addAttributePrefix("customer_interests", "bas");

            // Add an anything-but attribute
            fp.addAttributeAnythingBut("customer_interests", "baseball");

            // Add a filter policy attribute with a list of values
            ArrayList<String> attributeValues = new ArrayList<>();
            attributeValues.add("rugby");
            attributeValues.add("soccer");
            attributeValues.add("hockey");
            fp.addAttribute("customer_interests", attributeValues);

            // Add a numeric attribute
            fp.addAttribute("price_usd", "=", 0);

            // Add a numeric attribute with a range
            fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

            // Apply the filter policy attributes to an Amazon SNS subscription
            fp.apply(snsClient, subscriptionArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- APIの詳細については、「APIリファレンス[SetSubscriptionAttributes](#)」の「」を参照してください。AWS SDK for Java 2.x

## SetTopicAttributes

次の例は、SetTopicAttributes を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <attribute> <topicArn> <value>

            Where:
                attribute - The attribute action to use. Valid parameters are:
Policy | DisplayName | DeliveryPolicy .
                topicArn - The ARN of the topic.\s
                value - The value for the attribute.

            """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String attribute = args[0];
    String topicArn = args[1];
    String value = args[2];

    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    setTopAttr(snsClient, attribute, topicArn, value);
    snsClient.close();
}

public static void setTopAttr(SnsClient snsClient, String attribute, String
topicArn, String value) {
    try {
        SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
            .attributeName(attribute)
            .attributeValue(value)
            .topicArn(topicArn)
            .build();

        SetTopicAttributesResponse result =
snsClient.setTopicAttributes(request);
        System.out.println(
            "\n\nStatus was " + result.sdkHttpResponse().statusCode() + "\n
\nTopic " + request.topicArn()
                + " updated " + request.attributeName() + " to " +
request.attributeValue());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、「APIリファレンス[SetTopicAttributes](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## Subscribe

次の例は、Subscribe を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

E メールアドレスをトピックにサブスクライブします。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeEmail {
    public static void main(String[] args) {
        final String usage = ""
            Usage:    <topicArn> <email>

            Where:
                topicArn - The ARN of the topic to subscribe.
                email - The email address to use.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String topicArn = args[0];
String email = args[1];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .build();

subEmail(snsClient, topicArn, email);
snsClient.close();
}

public static void subEmail(SnsClient snsClient, String topicArn, String email)
{
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("email")
            .endpoint(email)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
            + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

HTTP エンドポイントをトピックにサブスクライブします。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn> <url>

            Where:
                topicArn - The ARN of the topic to subscribe.
                url - The HTTPS endpoint that you want to receive notifications.
            """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String url = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        subHTTPS(snsClient, topicArn, url);
        snsClient.close();
    }

    public static void subHTTPS(SnsClient snsClient, String topicArn, String url) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("https")
                .endpoint(url)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN is " + result.subscriptionArn() +
                "\n\n Status is ")
        }
    }
}
```

```
        + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

Lambda 関数をトピックにサブスクライブします。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <topicArn> <lambdaArn>

            Where:
                topicArn - The ARN of the topic to subscribe.
                lambdaArn - The ARN of an AWS Lambda function.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String topicArn = args[0];
String lambdaArn = args[1];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .build();

String arnValue = subLambda(snsClient, topicArn, lambdaArn);
System.out.println("Subscription ARN: " + arnValue);
snsClient.close();
}

public static String subLambda(SnsClient snsClient, String topicArn, String
lambdaArn) {
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("lambda")
            .endpoint(lambdaArn)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        return result.subscriptionArn();

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[Subscribe](#)」を参照してください。

## TagResource

次の例は、TagResource を使用する方法を説明しています。



## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddTags {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn>

            Where:
                topicArn - The ARN of the topic to which tags are added.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    addTopicTags(snsClient, topicArn);
    snsClient.close();
}

public static void addTopicTags(SnsClient snsClient, String topicArn) {
    try {
        Tag tag = Tag.builder()
            .key("Team")
            .value("Development")
            .build();

        Tag tag2 = Tag.builder()
            .key("Environment")
            .value("Gamma")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag);
        tagList.add(tag2);

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .resourceArn(topicArn)
            .tags(tagList)
            .build();

        snsClient.tagResource(tagResourceRequest);
        System.out.println("Tags have been added to " + topicArn);

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、「APIリファレンス[TagResource](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## Unsubscribe

次の例は、Unsubscribe を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <subscriptionArn>

            Where:
                subscriptionArn - The ARN of the subscription to delete.
            """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    unSub(snsClient, subscriptionArn);
    snsClient.close();
}

public static void unSub(SnsClient snsClient, String subscriptionArn) {
    try {
        UnsubscribeRequest request = UnsubscribeRequest.builder()
            .subscriptionArn(subscriptionArn)
            .build();

        UnsubscribeResponse result = snsClient.unsubscribe(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode()
            + "\n\nSubscription was removed for " +
request.subscriptionArn());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[Unsubscribe](#)」を参照してください。

## シナリオ

プッシュ通知のプラットフォームエンドポイントを作成します

次のコード例は、Amazon SNS プッシュ通知のプラットフォームエンドポイントを作成する方法を示しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = ""

                Usage:      <token> <platformApplicationArn>

                Where:
                    token - The name of the FIFO topic.\s
                    platformApplicationArn - The ARN value of platform application.
                You can get this value from the AWS Management Console.\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
CreatePlatformEndpointRequest.builder()
                .token(token)
                .platformApplicationArn(platformApplicationArn)
                .build();

            CreatePlatformEndpointResponse response =
snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

## FIFO トピックを作成して発行する

次のコード例は、FIFO Amazon SNS トピックを作成、発行する方法を示しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

この例では

- 1 つの Amazon SNS FIFO トピック、2 つの Amazon SQS FIFO キュー、および 1 つの標準キューを作成します。
- キューをトピックにサブスクライブし、メッセージをトピックに発行します。

[テスト](#)では、各キューへのメッセージの受信を検証します。[完全な例](#)では、アクセスポリシーの追加と、最後にリソースの削除も示しています。

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
            "Where:\n" +
            "    fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
            "    wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
            +
            "    retailQueueARN - The name of a SQS FIFO queue that will created
for the retail consumer. \n\n" +
            "    analyticsQueueARN - The name of a SQS standard queue that will
be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
```

```
final String wholeSaleQueueName = args[1];
final String retailQueueName = args[2];
final String analyticsQueueName = args[3];

// For convenience, the QueueData class holds metadata about a queue: ARN,
URL,
// name and type.
List<QueueData> queues = List.of(
    new QueueData(wholeSaleQueueName, QueueType.FIFO),
    new QueueData(retailQueueName, QueueType.FIFO),
    new QueueData(analyticsQueueName, QueueType.Standard));

// Create queues.
createQueues(queues);

// Create a topic.
String topicARN = createFIFOTopic(fifoTopicName);

// Subscribe each queue to the topic.
subscribeQueues(queues, topicARN);

// Allow the newly created topic to send messages to the queues.
addAccessPolicyToQueuesFINAL(queues, topicARN);

// Publish a sample price update message with payload.
publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

// Clean up resources.
deleteSubscriptions(queues);
deleteQueues(queues);
deleteTopic(topicARN);
}

public static String createFIFOTopic(String topicName) {
    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = Map.of(
            "FifoTopic", "true",
            "ContentBasedDeduplication", "false");

        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
            .name(topicName)
            .attributes(topicAttributes)
```



```
        .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        String topicArn = response.topicArn();
        System.out.println("The topic ARN is" + topicArn);

        return topicArn;

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void subscribeQueues(List<QueueData> queues, String topicARN) {
    queues.forEach(queue -> {
        SubscribeRequest subscribeRequest = SubscribeRequest.builder()
            .topicArn(topicARN)
            .endpoint(queue.queueARN)
            .protocol("sqs")
            .build();

        // Subscribe to the endpoint by using the SNS service client.
        // Only Amazon SQS queues can receive notifications from an Amazon SNS
        FIFO
        // topic.
        SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
        System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]);
        queue.subscriptionARN = subscribeResponse.subscriptionArn();
    });
}

public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {
    try {
        // Create and publish a message that updates the wholesale price.
        String subject = "Price Update";
        String dedupId = UUID.randomUUID().toString();
        String attributeName = "business";
        String attributeValue = "wholesale";
```

```
MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
    .dataType("String")
    .stringValue(attributeValue)
    .build();

Map<String, MessageAttributeValue> attributes = new HashMap<>();
attributes.put(attributeName, msgAttValue);
PublishRequest pubRequest = PublishRequest.builder()
    .topicArn(topicArn)
    .subject(subject)
    .message(payload)
    .messageGroupId(groupId)
    .messageDeduplicationId(dedupId)
    .messageAttributes(attributes)
    .build();

final PublishResponse response = snsClient.publish(pubRequest);
System.out.println(response.messageId());
System.out.println(response.sequenceNumber());
System.out.println("Message was published to " + topicArn);

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。

- [CreateTopic](#)
- [Publish](#)
- [Subscribe](#)

## トピックへの SMS メッセージの発行

次のコードの例は以下の方法を示しています。

- Amazon SNS トピックを作成します。
- 携帯電話番号をトピックにサブスクライブする。

- SMS メッセージをトピックに発行して、登録されているすべての電話番号がメッセージを一度に受信できるようにします。

## SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピックを作成し、その ARN を返します。

```
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 "";
}
}
```

トピックへのエンドポイントのサブスクライブ。

```
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);
    }
}
}
}

```

送信者の ID、上限価格、タイプなど、メッセージ上の属性を設定します。メッセージ属性はオプションです。

```

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);
    }
}
```

トピックへのメッセージの発行 メッセージは、すべての受信者に送信されます。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <message> <phoneNumber>

            Where:
```

```
        message - The message text to send.
        phoneNumber - The mobile phone number to which a message is sent
(for example, +1XXX5550100).\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String message = args[0];
    String phoneNumber = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();
    pubTextSMS(snsClient, message, phoneNumber);
    snsClient.close();
}

public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .phoneNumber(phoneNumber)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

## SMS テキストメッセージを発行する

次のコードサンプルは、Amazon SNS を使用して SMS メッセージを発行する方法を示しています。



## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <message> <phoneNumber>

                Where:
                    message - The message text to send.
                    phoneNumber - The mobile phone number to which a message is sent
(for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();
    pubTextSMS(snsClient, message, phoneNumber);
    snsClient.close();
}

public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .phoneNumber(phoneNumber)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[Publish](#)」を参照してください。

## サーバーレスサンプル

### Amazon SNS トリガーから Lambda 関数を呼び出す

次のコード例は、SNS トピックからメッセージを受信することによってトリガーされるイベントを受け取る Lambda 関数を実装する方法を示しています。この関数はイベントパラメータからメッセージを取得し、各メッセージの内容を記録します。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Java を使用した Lambda での SNS イベントの消費。

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

## SDK for Java 2.x を使用する Amazon SQS の例

次のコード例は、Amazon SQS AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### 開始方法

#### Hello Amazon SQS

次のコード例は、Amazon SQS の使用を開始する方法を示しています。

#### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「API リファレンス [ListQueues](#)」の「」を参照してください。AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)
- [サーバーレスサンプル](#)

## アクション

### CreateQueue

次の例は、CreateQueue を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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)
            .numberOfMessages(5)
            .build();
        return sqsClient.receiveMessage(receiveMessageRequest).messages();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }
    return null;
}

public static void changeMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {

    System.out.println("\nChange Message Visibility");
    try {

        for (Message message : messages) {
            ChangeMessageVisibilityRequest req =
ChangeMessageVisibilityRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .visibilityTimeout(100)
                .build();
            sqsClient.changeMessageVisibility(req);
        }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
    System.out.println("\nDelete Messages");

    try {
        for (Message message : messages) {
            DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .build();
            sqsClient.deleteMessage(deleteMessageRequest);
        }
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- API の詳細については、「API リファレンス [CreateQueue](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteMessage

次の例は、DeleteMessage を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
try {
    for (Message message : messages) {
        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
        sqsClient.deleteMessage(deleteMessageRequest);
    }
} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

- API の詳細については、「API リファレンス [DeleteMessage](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteQueue

次の例は、DeleteQueue を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteQueue {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <queueName>

            Where:
                queueName - The name of the Amazon SQS queue to delete.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        SqsClient sqs = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();
```

```
        deleteSQSQueue(sqs, queueName);
        sqs.close();
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();

            sqsClient.deleteQueue(deleteQueueRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「API リファレンス [DeleteQueue](#)」の「」を参照してください。AWS SDK for Java 2.x

## GetQueueUrl

次の例は、GetQueueUrl を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
GetQueueUrlResponse getQueueUrlResponse = sqsClient
    .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
    return getQueueUrlResponse.queueUrl();
```

- API の詳細については、「API リファレンス [GetQueueUrl](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListQueues

次の例は、ListQueues を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
}
```

- API の詳細については、「API リファレンス [ListQueues](#)」の「」を参照してください。AWS SDK for Java 2.x

## ReceiveMessage

次の例は、ReceiveMessage を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
try {
    ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .numberOfMessages(5)
        .build();
    return sqsClient.receiveMessage(receiveMessageRequest).messages();

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
```

- API の詳細については、「API リファレンス [ReceiveMessage](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## SendMessage

次の例は、SendMessage を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
}
}
```

- API の詳細については、「API リファレンス [SendMessage](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## SendMessageBatch

次の例は、SendMessageBatch を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
```

- API の詳細については、「API リファレンス [SendMessageBatch](#)」の「」を参照してください。AWS SDK for Java 2.x

## シナリオ

### FIFO トピックを作成して発行する

次のコード例は、FIFO Amazon SNS トピックを作成、発行する方法を示しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

この例では

- 1 つの Amazon SNS FIFO トピック、2 つの Amazon SQS FIFO キュー、および 1 つの標準キューを作成します。
- キューをトピックにサブスクライブし、メッセージをトピックに発行します。

[テスト](#)では、各キューへのメッセージの受信を検証します。[完全な例](#)では、アクセスポリシーの追加と、最後にリソースの削除も示しています。

```
public class PriceUpdateExample {
```

```
public final static SnsClient snsClient = SnsClient.create();
public final static SqsClient sqsClient = SqsClient.create();

public static void main(String[] args) {

    final String usage = "\n" +
        "Usage: " +
        "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
        "Where:\n" +
        "    fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
        "    wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
        +
        "    retailQueueARN - The name of a SQS FIFO queue that will created
for the retail consumer. \n\n" +
        "    analyticsQueueARN - The name of a SQS standard queue that will
be created for the analytics consumer. \n\n";
    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    final String fifoTopicName = args[0];
    final String wholeSaleQueueName = args[1];
    final String retailQueueName = args[2];
    final String analyticsQueueName = args[3];

    // For convenience, the QueueData class holds metadata about a queue: ARN,
URL,
    // name and type.
    List<QueueData> queues = List.of(
        new QueueData(wholeSaleQueueName, QueueType.FIFO),
        new QueueData(retailQueueName, QueueType.FIFO),
        new QueueData(analyticsQueueName, QueueType.Standard));

    // Create queues.
    createQueues(queues);

    // Create a topic.
    String topicARN = createFIFOTopic(fifoTopicName);

    // Subscribe each queue to the topic.
```

```
subscribeQueues(queues, topicARN);

// Allow the newly created topic to send messages to the queues.
addAccessPolicyToQueuesFINAL(queues, topicARN);

// Publish a sample price update message with payload.
publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

// Clean up resources.
deleteSubscriptions(queues);
deleteQueues(queues);
deleteTopic(topicARN);
}

public static String createFIFOtopic(String topicName) {
    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = Map.of(
            "FifoTopic", "true",
            "ContentBasedDeduplication", "false");

        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
            .name(topicName)
            .attributes(topicAttributes)
            .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        String topicArn = response.topicArn();
        System.out.println("The topic ARN is" + topicArn);

        return topicArn;
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void subscribeQueues(List<QueueData> queues, String topicARN) {
    queues.forEach(queue -> {
        SubscribeRequest subscribeRequest = SubscribeRequest.builder()
            .topicArn(topicARN)
```

```
        .endpoint(queue.queueARN)
        .protocol("sqs")
        .build();

    // Subscribe to the endpoint by using the SNS service client.
    // Only Amazon SQS queues can receive notifications from an Amazon SNS
FIFO
    // topic.
    SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
    System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]);
    queue.subscriptionARN = subscribeResponse.subscriptionArn();
    });
}

public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

    try {
        // Create and publish a message that updates the wholesale price.
        String subject = "Price Update";
        String dedupId = UUID.randomUUID().toString();
        String attributeName = "business";
        String attributeValue = "wholesale";

        MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
            .dataType("String")
            .stringValue(attributeValue)
            .build();

        Map<String, MessageAttributeValue> attributes = new HashMap<>();
        attributes.put(attributeName, msgAttValue);
        PublishRequest pubRequest = PublishRequest.builder()
            .topicArn(topicArn)
            .subject(subject)
            .message(payload)
            .messageGroupId(groupId)
            .messageDeduplicationId(dedupId)
            .messageAttributes(attributes)
            .build();

        final PublishResponse response = snsClient.publish(pubRequest);
        System.out.println(response.messageId());
    }
}
```

```
        System.out.println(response.sequenceNumber());
        System.out.println("Message was published to " + topicArn);

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [CreateTopic](#)
  - [Publish](#)
  - [Subscribe](#)

## サーバーレスサンプル

### Amazon SQS トリガーから Lambda 関数を呼び出す

次のコード例では、SQS キューからメッセージを受信することによってトリガーされるイベントを受け取る、Lambda 関数の実装方法を示しています。この関数はイベントパラメータからメッセージを取得し、各メッセージの内容を記録します。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [サーバーレスサンプル](#) リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Java を使用した Lambda での SQS イベントの消費。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
```

```
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }
    }
}
```

## Amazon SQS トリガーを使用した Lambda 関数でのバッチアイテムの失敗のレポート

以下のコード例では、SQS キューからイベントを受け取る Lambda 関数のための、部分的なバッチレスポンスの実装方法を示しています。この関数は、レスポンスとしてバッチアイテムの失敗を報告し、対象のメッセージを後で再試行するよう Lambda に伝えます。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [サーバーレスサンプル](#) リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Java を使用した Lambda での SQS バッチアイテム失敗のレポート。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {

        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
ArrayList<SQSBatchResponse.BatchItemFailure>();
        String messageId = "";
        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
                messageId = message.getMessageId();
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
SQSBatchResponse.BatchItemFailure(messageId));
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }
}
```

## SDK for Java 2.x を使用した Step Function の例

次のコード例は、Step Functions AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。



「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

## 開始方法

### Hello Step Functions

次のコード例は、Step Functions の使用を開始する方法を示しています。

### SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

## Java バージョンの 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);
        }
    }
}
```

- API の詳細については、「API リファレンス[ListStateMachines](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### CreateActivity

次の例は、CreateActivity を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String createActivity(SfnClient sfnClient, String activityName) {
    try {
        CreateActivityRequest activityRequest = CreateActivityRequest.builder()
            .name(activityName)
            .build();

        CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
        return response.activityArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- API の詳細については、「API リファレンス [CreateActivity](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateStateMachine

次の例は、CreateStateMachine を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
}
```

- API の詳細については、「API リファレンス [CreateStateMachine](#)」の「」を参照してください。AWS SDK for Java 2.x

## DeleteActivity

次の例は、DeleteActivity を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
}
}
```

- API の詳細については、「API リファレンス [DeleteActivity](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteStateMachine

次の例は、DeleteStateMachine を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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.");
}
```

- API の詳細については、「API リファレンス [DeleteStateMachine](#)」の「」を参照してください。AWS SDK for Java 2.x

## DescribeExecution

次の例は、DescribeExecution を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
}
}
```

- API の詳細については、「API リファレンス [DescribeExecution](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DescribeStateMachine

次の例は、DescribeStateMachine を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例  
を見つけて、設定と実行の方法を確認してください。

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

- API の詳細については、「API リファレンス [DescribeStateMachine](#)」の「」を参照してください。AWS SDK for Java 2.x

## GetActivityTask

次の例は、GetActivityTask を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
    List<String> myList = new ArrayList<>();
    GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
```



```
        .activityArn(actArn)
        .build();

    GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
    myList.add(response.taskToken());
    myList.add(response.input());
    return myList;
}

/// <summary>
/// Stop execution of a Step Functions workflow.
/// </summary>
/// <param name="executionArn">The Amazon Resource Name (ARN) of
/// the Step Functions execution to stop.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> StopExecution(string executionArn)
{
    var response =
        await _amazonStepFunctions.StopExecutionAsync(new StopExecutionRequest
{ ExecutionArn = executionArn });
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- APIの詳細については、「APIリファレンス[GetActivityTask](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListActivities

次の例は、ListActivities を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [ListActivities](#)」の「」を参照してください。AWS SDK for Java 2.x

## ListExecutions

次の例は、ListExecutions を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
```

```
}
```

- API の詳細については、「API リファレンス [ListExecutions](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## ListStateMachines

次の例は、ListStateMachines を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「API リファレンス [ListStateMachines](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## SendTaskSuccess

次の例は、SendTaskSuccess を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
    try {
        SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
```

```
        .taskToken(token)
        .output(json)
        .build();

    sfmClient.sendTaskSuccess(successRequest);

} catch (SfmException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス [SendTaskSuccess](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## StartExecution

次の例は、StartExecution を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String startWorkflow(SfmClient sfmClient, 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 =
            sfmClient.startExecution(executionRequest);
    }
```

```
        return response.executionArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- API の詳細については、「API リファレンス [StartExecution](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## シナリオ

### ステートマシンの使用を開始する

次のコードサンプルは、以下の操作方法を示しています。

- アクティビティを作成します。
- 以前に作成したアクティビティをステップとして含む Amazon ステート言語定義からステートマシンを作成します。
- ステートマシンを実行し、ユーザー入力でアクティビティに応答します。
- 実行が完了したら最終ステータスと出力を取得して、リソースをクリーンアップします。

### SDK for Java 2.x

#### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * You can obtain the JSON file to create a state machine in the following
 * GitHub location.
 *
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files
 */
```

```
* To run this code example, place the chat_sfn_state_machine.json file into
* your project's resources folder.
*
* Also, set up your development environment, including your credentials.
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This Java code example performs the following tasks:
*
* 1. Creates an activity.
* 2. Creates a state machine.
* 3. Describes the state machine.
* 4. Starts execution of the state machine and interacts with it.
* 5. Describes the execution.
* 6. Delete the activity.
* 7. Deletes the state machine.
*/
public class StepFunctionsScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws Exception {
        final String usage = ""

            Usage:
                <roleARN> <activityName> <stateMachineName>

            Where:
                roleName - The name of the IAM role to create for this state
machine.

                activityName - The name of an activity to create.
                stateMachineName - The name of the state machine to create.

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String activityName = args[1];
        String stateMachineName = args[2];
        String polJSON = "{\n" +
```



```

    "    \"Version\": \"2012-10-17\", \"n\" +
    "    \"Statement\": [\"n\" +
    "        {\"n\" +
    "            \"Sid\": \"\", \"n\" +
    "            \"Effect\": \"Allow\", \"n\" +
    "            \"Principal\": {\"n\" +
    "                \"Service\": \"states.amazonaws.com\", \"n\" +
    "            }, \"n\" +
    "            \"Action\": \"sts:AssumeRole\", \"n\" +
    "        } \"n\" +
    "    ] \"n\" +
    "};

```

```

Scanner sc = new Scanner(System.in);
boolean action = false;

```

```

Region region = Region.US_EAST_1;
SfnClient sfnClient = SfnClient.builder()
    .region(region)
    .build();

```

```

Region regionGl = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(regionGl)
    .build();

```

```

System.out.println(DASHES);
System.out.println("Welcome to the AWS Step Functions example scenario.");
System.out.println(DASHES);

```

```

System.out.println(DASHES);
System.out.println("1. Create an activity.");
String activityArn = createActivity(sfnClient, activityName);
System.out.println("The ARN of the activity is " + activityArn);
System.out.println(DASHES);

```

```

// Get JSON to use for the state machine and place the activityArn value
into

```

```

// it.
InputStream input = StepFunctionsScenario.class.getClassLoader()
    .getResourceAsStream("chat_sfn_state_machine.json");
ObjectMapper mapper = new ObjectMapper();
JsonNode jsonNode = mapper.readValue(input, JsonNode.class);
String jsonString = mapper.writeValueAsString(jsonNode);

```

```
// Modify the Resource node.
ObjectMapper objectMapper = new ObjectMapper();
JsonNode root = objectMapper.readTree(jsonString);
((ObjectNode) root.path("States").path("GetInput")).put("Resource",
activityArn);

// Convert the modified Java object back to a JSON string.
String stateDefinition = objectMapper.writeValueAsString(root);
System.out.println(stateDefinition);

System.out.println(DASHES);
System.out.println("2. Create a state machine.");
String roleARN = createIAMRole(iam, roleName, polJSON);
String stateMachineArn = createMachine(sfnClient, roleARN, stateMachineName,
stateDefinition);
System.out.println("The ARN of the state machine is " + stateMachineArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Describe the state machine.");
describeStateMachine(sfnClient, stateMachineArn);
System.out.println("What should ChatSFN call you?");
String userName = sc.nextLine();
System.out.println("Hello " + userName);
System.out.println(DASHES);

System.out.println(DASHES);
// The JSON to pass to the StartExecution call.
String executionJson = "{ \"name\" : \"" + userName + "\" }";
System.out.println(executionJson);
System.out.println("4. Start execution of the state machine and interact
with it.");
String runArn = startWorkflow(sfnClient, stateMachineArn, executionJson);
System.out.println("The ARN of the state machine execution is " + runArn);
List<String> myList;
while (!action) {
    myList = getActivityTask(sfnClient, activityArn);
    System.out.println("ChatSFN: " + myList.get(1));
    System.out.println(userName + " please specify a value.");
    String myAction = sc.nextLine();
    if (myAction.compareTo("done") == 0)
        action = true;
}
```

```
        System.out.println("You have selected " + myAction);
        String taskJson = "{ \"action\" : \"" + myAction + "\" }";
        System.out.println(taskJson);
        sendTaskSuccess(sfnClient, myList.get(0), taskJson);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("5. Describe the execution.");
    describeExe(sfnClient, runArn);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("6. Delete the activity.");
    deleteActivity(sfnClient, activityArn);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("7. Delete the state machines.");
    deleteMachine(sfnClient, stateMachineArn);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The AWS Step Functions example scenario is complete.");
    System.out.println(DASHES);
}

public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(polJSON)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        return response.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

```
}

public static void describeExe(SfnClient sfnClient, String executionArn) {
    try {
        DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
            .executionArn(executionArn)
            .build();

        String status = "";
        boolean hasSucceeded = false;
        while (!hasSucceeded) {
            DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
            status = response.statusAsString();
            if (status.compareTo("RUNNING") == 0) {
                System.out.println("The state machine is still running, let's
wait for it to finish.");
                Thread.sleep(2000);
            } else if (status.compareTo("SUCCEEDED") == 0) {
                System.out.println("The Step Function workflow has succeeded");
                hasSucceeded = true;
            } else {
                System.out.println("The Status is neither running or
succeeded");
            }
        }
        System.out.println("The Status is " + status);

    } catch (SfnException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
    try {
        SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
            .taskToken(token)
            .output(json)
            .build();

        sfnClient.sendTaskSuccess(successRequest);
    }
}
```

```
    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<String> getActivityTask(SfnClient sfncClient, String actArn) {
    List<String> myList = new ArrayList<>();
    GetActivityTaskRequest getActivityTaskRequest =
    GetActivityTaskRequest.builder()
        .activityArn(actArn)
        .build();

    GetActivityTaskResponse response =
    sfncClient.getActivityTask(getActivityTaskRequest);
    myList.add(response.taskToken());
    myList.add(response.input());
    return myList;
}

public static void deleteActivity(SfnClient sfncClient, String actArn) {
    try {
        DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
            .activityArn(actArn)
            .build();

        sfncClient.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 sfncClient, 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 =
            sfmClient.createActivity(activityRequest);
        return response.activityArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [CreateActivity](#)
  - [CreateStateMachine](#)
  - [DeleteActivity](#)
  - [DeleteStateMachine](#)
  - [DescribeExecution](#)
  - [DescribeStateMachine](#)
  - [GetActivityTask](#)
  - [ListActivities](#)
  - [ListStateMachines](#)
  - [SendTaskSuccess](#)
  - [StartExecution](#)
  - [StopExecution](#)

## AWS STS SDK for Java 2.x を使用した の例

次のコード例は、AWS SDK for Java 2.x でを使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS STS。



アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック


- [アクション](#)

## アクション

### AssumeRole

次の例は、AssumeRole を使用する方法を説明しています。

SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
```

```

* Then define a Trust Relationship in the AWS Console. You can use this as an
* example:
*
* {
*   "Version": "2012-10-17",
*   "Statement": [
*     {
*       "Effect": "Allow",
*       "Principal": {
*         "AWS": "<Specify the ARN of your IAM user you are using in this code
*         example>"
*       },
*       "Action": "sts:AssumeRole"
*     }
*   ]
* }
*
* For more information, see "Editing the Trust Relationship for an Existing
* Role" in the AWS Directory Service guide.
*
* Also, set up your development environment, including your credentials.
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <roleArn> <roleSessionName>\s

                Where:
                roleArn - The Amazon Resource Name (ARN) of the role to assume
(for example, rn:aws:iam::000008047983:role/s3role).\s
                roleSessionName - An identifier for the assumed role session
(for example, mysession).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}

```

```
String roleArn = args[0];
String roleSessionName = args[1];
Region region = Region.US_EAST_1;
StsClient stsClient = StsClient.builder()
    .region(region)
    .build();

assumeGivenRole(stsClient, roleArn, roleSessionName);
stsClient.close();
}

public static void assumeGivenRole(StsClient stsClient, String roleArn, String
roleSessionName) {
    try {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
            .roleArn(roleArn)
            .roleSessionName(roleSessionName)
            .build();

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        Credentials myCreds = roleResponse.credentials();

        // Display the time when the temp creds expire.
        Instant exTime = myCreds.expiration();
        String tokenInfo = myCreds.sessionToken();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(exTime);
        System.out.println("The token " + tokenInfo + " expires on " + exTime);

    } catch (StsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- API の詳細については、「API リファレンス [AssumeRole](#)」の「」を参照してください。AWS SDK for Java 2.x

## AWS Support SDK for Java 2.x を使用した の例

次のコード例は、AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS Support。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

### 開始方法

こんにちは AWS Supportは

次のコード例は、AWS Supportの使用を開始する方法を示しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SupportException;
import java.util.ArrayList;
import java.util.List;
```

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following task:
 *
 * 1. Gets and displays available services.
 *
 * NOTE: To see multiple operations, see SupportScenario.
 */

public class HelloSupport {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
            .region(region)
            .build();

        System.out.println("***** Step 1. Get and display available services.");
        displayServices(supportClient);
    }

    // Return a List that contains a Service name and Category name.
    public static void displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
                DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
                supportClient.describeServices(servicesRequest);
            List<Service> services = response.services();
        }
    }
}
```

```
System.out.println("Get the first 10 services");
int index = 1;
for (Service service : services) {
    if (index == 11)
        break;

    System.out.println("The Service name is: " + service.name());

    // Display the Categories for this service.
    List<Category> categories = service.categories();
    for (Category cat : categories) {
        System.out.println("The category name is: " + cat.name());
    }
    index++;
}

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス [DescribeServices](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### AddAttachmentsToSet

次の例は、AddAttachmentsToSet を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
    try {
        File myFile = new File(fileAttachment);
        InputStream sourceStream = new FileInputStream(myFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        Attachment attachment = Attachment.builder()
            .fileName(myFile.getName())
            .data(sourceBytes)
            .build();

        AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
            .attachments(attachment)
            .build();

        AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
        return response.attachmentSetId();

    } catch (SupportException | FileNotFoundException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- API の詳細については、「API リファレンス [AddAttachmentsToSet](#)」の「」を参照してください。 AWS SDK for Java 2.x

## AddCommunicationToCase

次の例は、AddCommunicationToCase を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
    try {
        AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
            .caseId(caseId)
            .attachmentSetId(attachmentSetId)
            .communicationBody("Please refer to attachment for details.")
            .build();

        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
        if (response.result())
            System.out.println("You have successfully added a communication to
an AWS Support case");
        else
            System.out.println("There was an error adding the communication to
an AWS Support case");

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [AddCommunicationToCase](#)」の「」を参照してください。 AWS SDK for Java 2.x



## CreateCase

次の例は、CreateCase を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
    try {
        String serviceCode = sevCatList.get(0);
        String caseCat = sevCatList.get(1);
        CreateCaseRequest caseRequest = CreateCaseRequest.builder()
            .categoryCode(caseCat.toLowerCase())
            .serviceCode(serviceCode.toLowerCase())
            .severityCode(sevLevel.toLowerCase())
            .communicationBody("Test issue with " +
serviceCode.toLowerCase())
            .subject("Test case, please ignore")
            .language("en")
            .issueType("technical")
            .build();

        CreateCaseResponse response = supportClient.createCase(caseRequest);
        return response.caseId();

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- API の詳細については、「API リファレンス [CreateCase](#)」の「」を参照してください。 AWS SDK for Java 2.x

## DescribeAttachment

次の例は、DescribeAttachment を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void describeAttachment(SupportClient supportClient, String
attachId) {
    try {
        DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
            .attachmentId(attachId)
            .build();

        DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
        System.out.println("The name of the file is " +
response.attachment().fileName());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DescribeAttachment](#)」の「」を参照してください。 AWS SDK for Java 2.x

## DescribeCases

次の例は、DescribeCases を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void getOpenCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
            .maxResults(20)
            .afterTime(yesterday.toString())
            .beforeTime(now.toString())
            .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
            System.out.println("The case status is " + sinCase.status());
            System.out.println("The case Id is " + sinCase.caseId());
            System.out.println("The case subject is " + sinCase.subject());
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DescribeCases](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DescribeCommunications

次の例は、DescribeCommunications を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String listCommunications(SupportClient supportClient, String
caseId) {
    try {
        String attachId = null;
        DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
            .caseId(caseId)
            .maxResults(10)
            .build();

        DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
        List<Communication> communications = response.communications();
        for (Communication comm : communications) {
            System.out.println("the body is: " + comm.body());

            // Get the attachment id value.
            List<AttachmentDetails> attachments = comm.attachmentSet();
            for (AttachmentDetails detail : attachments) {
                attachId = detail.attachmentId();
            }
        }
        return attachId;

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- API の詳細については、「API リファレンス [DescribeCommunications](#)」の「」を参照してください。AWS SDK for Java 2.x

## DescribeServices

次の例は、DescribeServices を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
            .language("en")
            .build();

        DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
        String serviceCode = null;
        String catName = null;
        List<String> sevCatList = new ArrayList<>();
        List<Service> services = response.services();

        System.out.println("Get the first 10 services");
        int index = 1;
        for (Service service : services) {
            if (index == 11)
                break;

            System.out.println("The Service name is: " + service.name());
            if (service.name().compareTo("Account") == 0)
                serviceCode = service.code();
        }
    }
}
```

```
        // Get the Categories for this service.
        List<Category> categories = service.categories();
        for (Category cat : categories) {
            System.out.println("The category name is: " + cat.name());
            if (cat.name().compareTo("Security") == 0)
                catName = cat.name();
        }
        index++;
    }

    // Push the two values to the list.
    sevCatList.add(serviceCode);
    sevCatList.add(catName);
    return sevCatList;

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return null;
}
```

- API の詳細については、「API リファレンス [DescribeServices](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DescribeSeverityLevels

次の例は、DescribeSeverityLevels を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください [GitHub](#)。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String displaySevLevels(SupportClient supportClient) {
    try {
```

```
DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
    .language("en")
    .build();

DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
List<SeverityLevel> severityLevels = response.severityLevels();
String levelName = null;
for (SeverityLevel sevLevel : severityLevels) {
    System.out.println("The severity level name is: " +
sevLevel.name());
    if (sevLevel.name().compareTo("High") == 0)
        levelName = sevLevel.name();
}
return levelName;

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return "";
}
```

- API の詳細については、「API リファレンス[DescribeSeverityLevels](#)」の「」を参照してください。AWS SDK for Java 2.x

## ResolveCase

次の例は、ResolveCase を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
            .caseId(caseId)
            .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
        System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [ResolveCase](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## シナリオ

### ケースを開始する

次のコードサンプルは、以下の操作方法を示しています。

- ケースの利用可能なサービスと重要度レベルを取得して表示する方法
- 選択したサービス、カテゴリ、重要度レベルを使用してサポートケースを作成する方法
- 当日のオープンケースのリストを取得して表示する方法
- 新しいケースに添付セットとコミュニケーションを追加する方法
- ケースの新しい添付ファイルとコミュニケーションについて説明する方法
- ケースを解決する方法
- 当日の解決済みケースのリストを取得して表示する方法



## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

さまざまな AWS Support オペレーションを実行します。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetResponse;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseRequest;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseResponse;
import software.amazon.awssdk.services.support.model.Attachment;
import software.amazon.awssdk.services.support.model.AttachmentDetails;
import software.amazon.awssdk.services.support.model.CaseDetails;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.Communication;
import software.amazon.awssdk.services.support.model.CreateCaseRequest;
import software.amazon.awssdk.services.support.model.CreateCaseResponse;
import software.amazon.awssdk.services.support.model.DescribeAttachmentRequest;
import software.amazon.awssdk.services.support.model.DescribeAttachmentResponse;
import software.amazon.awssdk.services.support.model.DescribeCasesRequest;
import software.amazon.awssdk.services.support.model.DescribeCasesResponse;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsRequest;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsResponse;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsRequest;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsResponse;
import software.amazon.awssdk.services.support.model.ResolveCaseRequest;
import software.amazon.awssdk.services.support.model.ResolveCaseResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SeverityLevel;
import software.amazon.awssdk.services.support.model.SupportException;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetRequest;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
```

```
import java.time.Instant;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets and displays available services.
 * 2. Gets and displays severity levels.
 * 3. Creates a support case by using the selected service, category, and
 * severity level.
 * 4. Gets a list of open cases for the current day.
 * 5. Creates an attachment set with a generated file.
 * 6. Adds a communication with the attachment to the support case.
 * 7. Lists the communications of the support case.
 * 8. Describes the attachment set included with the communication.
 * 9. Resolves the support case.
 * 10. Gets a list of resolved cases for the current day.
 */
public class SupportScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <fileAttachment>Where:
            fileAttachment - The file can be a simple saved .txt file to use
            as an email attachment.\s
            """;
    }
}
```

```
    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String fileAttachment = args[0];
    Region region = Region.US_WEST_2;
    SupportClient supportClient = SupportClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("***** Welcome to the AWS Support case example
scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Get and display available services.");
    List<String> sevCatList = displayServices(supportClient);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Get and display Support severity levels.");
    String sevLevel = displaySevLevels(supportClient);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Create a support case using the selected service,
category, and severity level.");
    String caseId = createSupportCase(supportClient, sevCatList, sevLevel);
    if (caseId.compareTo("") == 0) {
        System.out.println("A support case was not successfully created!");
        System.exit(1);
    } else
        System.out.println("Support case " + caseId + " was successfully
created!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Get open support cases.");
    getOpenCase(supportClient);
    System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("5. Create an attachment set with a generated file to add
to the case.");
        String attachmentSetId = addAttachment(supportClient, fileAttachment);
        System.out.println("The Attachment Set id value is" + attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add communication with the attachment to the support
case.");
        addAttachSupportCase(supportClient, caseId, attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. List the communications of the support case.");
        String attachId = listCommunications(supportClient, caseId);
        System.out.println("The Attachment id value is" + attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Describe the attachment set included with the
communication.");
        describeAttachment(supportClient, attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Resolve the support case.");
        resolveSupportCase(supportClient, caseId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of resolved cases for the current day.");
        getResolvedCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("***** This Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static void getResolvedCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
```

```
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
        .maxResults(30)
        .afterTime(yesterday.toString())
        .beforeTime(now.toString())
        .includeResolvedCases(true)
        .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
            if (sinCase.status().compareTo("resolved") == 0)
                System.out.println("The case status is " + sinCase.status());
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
            .caseId(caseId)
            .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
        System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeAttachment(SupportClient supportClient, String
attachId) {
```

```
        try {
            DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
                .attachmentId(attachId)
                .build();

            DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
            System.out.println("The name of the file is " +
response.attachment().fileName());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String listCommunications(SupportClient supportClient, String
caseId) {
        try {
            String attachId = null;
            DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
                .caseId(caseId)
                .maxResults(10)
                .build();

            DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
            List<Communication> communications = response.communications();
            for (Communication comm : communications) {
                System.out.println("the body is: " + comm.body());

                // Get the attachment id value.
                List<AttachmentDetails> attachments = comm.attachmentSet();
                for (AttachmentDetails detail : attachments) {
                    attachId = detail.attachmentId();
                }
            }
            return attachId;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

```
    }
    return "";
}

public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
    try {
        AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
            .caseId(caseId)
            .attachmentSetId(attachmentSetId)
            .communicationBody("Please refer to attachment for details.")
            .build();

        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
        if (response.result())
            System.out.println("You have successfully added a communication to
an AWS Support case");
        else
            System.out.println("There was an error adding the communication to
an AWS Support case");

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
    try {
        File myFile = new File(fileAttachment);
        InputStream sourceStream = new FileInputStream(myFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        Attachment attachment = Attachment.builder()
            .fileName(myFile.getName())
            .data(sourceBytes)
            .build();

        AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
            .attachments(attachment)
```

```
        .build();

        AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
        return response.attachmentSetId();

    } catch (SupportException | FileNotFoundException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static void getOpenCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
            .maxResults(20)
            .afterTime(yesterday.toString())
            .beforeTime(now.toString())
            .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
            System.out.println("The case status is " + sinCase.status());
            System.out.println("The case Id is " + sinCase.caseId());
            System.out.println("The case subject is " + sinCase.subject());
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
```



```
try {
    String serviceCode = sevCatList.get(0);
    String caseCat = sevCatList.get(1);
    CreateCaseRequest caseRequest = CreateCaseRequest.builder()
        .categoryCode(caseCat.toLowerCase())
        .serviceCode(serviceCode.toLowerCase())
        .severityCode(sevLevel.toLowerCase())
        .communicationBody("Test issue with " +
serviceCode.toLowerCase())
        .subject("Test case, please ignore")
        .language("en")
        .issueType("technical")
        .build();

    CreateCaseResponse response = supportClient.createCase(caseRequest);
    return response.caseId();

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return "";
}

public static String displaySevLevels(SupportClient supportClient) {
    try {
        DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
            .language("en")
            .build();

        DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
        List<SeverityLevel> severityLevels = response.severityLevels();
        String levelName = null;
        for (SeverityLevel sevLevel : severityLevels) {
            System.out.println("The severity level name is: " +
sevLevel.name());
            if (sevLevel.name().compareTo("High") == 0)
                levelName = sevLevel.name();
        }
        return levelName;

    } catch (SupportException e) {
```

```
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
            .language("en")
            .build();

        DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
        String serviceCode = null;
        String catName = null;
        List<String> sevCatList = new ArrayList<>();
        List<Service> services = response.services();

        System.out.println("Get the first 10 services");
        int index = 1;
        for (Service service : services) {
            if (index == 11)
                break;

            System.out.println("The Service name is: " + service.name());
            if (service.name().compareTo("Account") == 0)
                serviceCode = service.code();

            // Get the Categories for this service.
            List<Category> categories = service.categories();
            for (Category cat : categories) {
                System.out.println("The category name is: " + cat.name());
                if (cat.name().compareTo("Security") == 0)
                    catName = cat.name();
            }
            index++;
        }

        // Push the two values to the list.
        sevCatList.add(serviceCode);
        sevCatList.add(catName);
    }
}
```

```
        return sevCatList;

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return null;
}
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [AddAttachmentsToSet](#)
  - [AddCommunicationToCase](#)
  - [CreateCase](#)
  - [DescribeAttachment](#)
  - [DescribeCases](#)
  - [DescribeCommunications](#)
  - [DescribeServices](#)
  - [DescribeSeverityLevels](#)
  - [ResolveCase](#)

## SDK for Java 2.x を使用した Systems Manager の例

次のコード例は、Systems Manager AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

## 開始方法

こんにちは、Systems Manager

次のコード例は、Systems Manager の使用を開始する方法を示しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.DocumentFilter;
import software.amazon.awssdk.services.ssm.model.ListDocumentsRequest;
import software.amazon.awssdk.services.ssm.model.ListDocumentsResponse;

public class HelloSSM {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <awsAccount>

            Where:
                awsAccount - Your AWS Account number.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String awsAccount = args[0] ;
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();
```

```
        listDocuments(ssmClient, awsAccount);
    }

    /**
     * This code automatically fetches the next set of results using the `nextToken`
     * and
     * stops once the desired maxResults (20 in this case) have been reached.
     */
    public static void listDocuments(SsmClient ssmClient, String awsAccount) {
        String nextToken = null;
        int totalDocumentsReturned = 0;
        int maxResults = 20;
        do {
            ListDocumentsRequest request = ListDocumentsRequest.builder()
                .documentFilterList(
                    DocumentFilter.builder()
                        .key("Owner")
                        .value(awsAccount)
                        .build()
                )
                .maxResults(maxResults)
                .nextToken(nextToken)
                .build();

            ListDocumentsResponse response = ssmClient.listDocuments(request);
            response.documentIdentifiers().forEach(identifier ->
                System.out.println("Document Name: " + identifier.name()));
            nextToken = response.nextToken();
            totalDocumentsReturned += response.documentIdentifiers().size();
        } while (nextToken != null && totalDocumentsReturned < maxResults);
    }
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[listThings](#)」を参照してください。

## トピック

- [アクション](#)
- [シナリオ](#)

## アクション

### CreateDocument

次の例は、CreateDocument を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Create an AWS SSM document to use in this scenario.
public static void createSSMDoc(SsmClient ssmClient, String docName) {
    // Create JSON for the content
    String jsonData = ""
        {
            "schemaVersion": "2.2",
            "description": "Run a simple shell command",
            "mainSteps": [
                {
                    "action": "aws:runShellScript",
                    "name": "runEchoCommand",
                    "inputs": {
                        "runCommand": [
                            "echo 'Hello, world!'"
                        ]
                    }
                }
            ]
        }
        """;

    try {
        CreateDocumentRequest request = CreateDocumentRequest.builder()
            .content(jsonData)
            .name(docName)
            .documentType(DocumentType.COMMAND)
            .build();

        // Create the document.
```

```
        CreateDocumentResponse response = ssmClient.createDocument(request);
        System.out.println("The status of the document is " +
response.documentDescription().status());

    } catch (DocumentAlreadyExistsException e) {
        System.err.println("The document already exists. Moving on." );
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [CreateDocument](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## CreateMaintenanceWindow

次の例は、CreateMaintenanceWindow を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static String createMaintenanceWindow(SsmClient ssmClient, String
winName) {
    CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
        .name(winName)
        .description("This is my maintenance window")
        .allowUnassociatedTargets(true)
        .duration(2)
        .cutoff(1)
        .schedule("cron(0 10 ? * MON-FRI *)")
        .build();

    try {
```

```
        CreateMaintenanceWindowResponse response =
ssmClient.createMaintenanceWindow(request);
        String maintenanceWindowId = response.windowId();
        System.out.println("The maintenance window id is " +
maintenanceWindowId);
        return maintenanceWindowId;

    } catch (DocumentAlreadyExistsException e) {
        System.err.println("The maintenance window already exists. Moving on.");
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
        .key("name")
        .values(winName)
        .build();

    DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
        .filters(filter)
        .build();

    String windowId = "";
    DescribeMaintenanceWindowsResponse response =
ssmClient.describeMaintenanceWindows(winRequest);
    List<MaintenanceWindowIdentity> windows = response.windowIdentities();
    if (!windows.isEmpty()) {
        windowId = windows.get(0).windowId();
        System.out.println("Window ID: " + windowId);
    } else {
        System.out.println("Window not found.");
    }
    return windowId;
}
```

- APIの詳細については、「APIリファレンス[CreateMaintenanceWindow](#)」の「」を参照してください。AWS SDK for Java 2.x



## CreateOpsItem

次の例は、CreateOpsItem を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Create an SSM OpsItem
public static String createSSMOpsItem(SsmClient ssmClient, String title, String
source, String category, String severity) {
    try {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
            .description("Created by the Systems Manager Java API")
            .title(title)
            .source(source)
            .category(category)
            .severity(severity)
            .build();

        CreateOpsItemResponse itemResponse =
ssmClient.createOpsItem(opsItemRequest);
        return itemResponse.opsItemId();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- API の詳細については、「API リファレンス [CreateOpsItem](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteDocument

次の例は、DeleteDocument を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Deletes an AWS Systems Manager document.
public static void deleteDoc(SsmClient ssmClient, String documentName) {
    try {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
            .name(documentName)
            .build();

        ssmClient.deleteDocument(documentRequest);
        System.out.println("The Systems Manager document was successfully
deleted.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DeleteDocument](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DeleteMaintenanceWindow

次の例は、DeleteMaintenanceWindow を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void deleteMaintenanceWindow(SsmClient ssmClient, String winId) {
    try {
        DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
        .windowId(winId)
        .build();

        ssmClient.deleteMaintenanceWindow(windowRequest);
        System.out.println("The maintenance window was successfully deleted.");


    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DeleteMaintenanceWindow](#)」の「」を参照してください。 AWS SDK for Java 2.x

## DescribeOpsItems

次の例は、DescribeOpsItems を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void describeOpsItems(SsmClient ssmClient, String key) {
    try {
        OpsItemFilter filter = OpsItemFilter.builder()
            .key(OpsItemFilterKey.OPS_ITEM_ID)
            .values(key)
            .operator(OpsItemFilterOperator.EQUAL)
            .build();

        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
            .maxResults(10)
            .opsItemFilters(filter)
            .build();

        DescribeOpsItemsResponse itemsResponse =
            ssmClient.describeOpsItems(itemsRequest);
        List<OpsItemSummary> items = itemsResponse.opsItemSummaries();
        for (OpsItemSummary item : items) {
            System.out.println("The item title is " + item.title() + " and the
status is "+item.status().toString());
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [DescribeOpsItems](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DescribeParameters

次の例は、DescribeParameters を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [DescribeParameters](#)」の「」を参照してください。AWS SDK for Java 2.x

## PutParameter

次の例は、PutParameter を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [PutParameter](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## SendCommand

次の例は、SendCommand を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Sends a SSM command to a managed node.
public static String sendSSMCommand(SsmClient ssmClient, String documentName,
String instanceId) throws InterruptedException {
    // Before we use Document to send a command - make sure it is active.
    boolean isDocumentActive = false;
    DescribeDocumentRequest request = DescribeDocumentRequest.builder()
        .name(documentName)
        .build();

    while (!isDocumentActive) {
        DescribeDocumentResponse response = ssmClient.describeDocument(request);
        String documentStatus = response.document().statusAsString();
        if (documentStatus.equals("Active")) {
            System.out.println("The Systems Manager document is active and ready
to use.");
            isDocumentActive = true;
        } else {
            System.out.println("The Systems Manager document is not active.
Status: " + documentStatus);
            try {
                // Add a delay to avoid making too many requests.
                Thread.sleep(5000); // Wait for 5 seconds before checking again
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```



```
    }
  }
}

// Create the SendCommandRequest.
SendCommandRequest commandRequest = SendCommandRequest.builder()
    .documentName(documentName)
    .instanceIds(instanceId)
    .build();

// Send the command.
SendCommandResponse commandResponse = ssmClient.sendCommand(commandRequest);
String commandId = commandResponse.command().commandId();
System.out.println("The command Id is " + commandId);

// Wait for the command execution to complete.
GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
    .commandId(commandId)
    .instanceId(instanceId)
    .build();

System.out.println("Wait 5 secs");
TimeUnit.SECONDS.sleep(5);

// Retrieve the command execution details.
GetCommandInvocationResponse commandInvocationResponse =
ssmClient.getCommandInvocation(invocationRequest);

// Check the status of the command execution.
CommandInvocationStatus status = commandInvocationResponse.status();
if (status == CommandInvocationStatus.SUCCESS) {
    System.out.println("Command execution successful.");
} else {
    System.out.println("Command execution failed. Status: " + status);
}
return commandId;
}
```

- APIの詳細については、「APIリファレンス[SendCommand](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## UpdateMaintenanceWindow

次の例は、UpdateMaintenanceWindow を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Update the maintenance window schedule
public static void updateSSMMaintenanceWindow(SsmClient ssmClient, String id,
String name) {
    try {
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
            .windowId(id)
            .allowUnassociatedTargets(true)
            .duration(24)
            .enabled(true)
            .name(name)
            .schedule("cron(0 0 ? * MON *)")
            .build();

        ssmClient.updateMaintenanceWindow(updateRequest);
        System.out.println("The Systems Manager maintenance window was
successfully updated.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [UpdateMaintenanceWindow](#)」の「」を参照してください。 AWS SDK for Java 2.x

## UpdateOpsItem

次の例は、UpdateOpsItem を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static void resolveOpsItem(SsmClient ssmClient, String opsID) {
    try {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
            .opsItemId(opsID)
            .status(OpsItemStatus.RESOLVED)
            .build();

        ssmClient.updateOpsItem(opsItemRequest);

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「API リファレンス [UpdateOpsItem](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## シナリオ

### Systems Manager の使用を開始する

次のコード例は、Systems Manager のメンテナンスウィンドウ、ドキュメント、および を使用する  
方法を示しています OpsItems。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.CommandInvocation;
import software.amazon.awssdk.services.ssm.model.CommandInvocationStatus;
import software.amazon.awssdk.services.ssm.model.CreateDocumentRequest;
import software.amazon.awssdk.services.ssm.model.CreateDocumentResponse;
import software.amazon.awssdk.services.ssm.model.CreateMaintenanceWindowRequest;
import software.amazon.awssdk.services.ssm.model.CreateMaintenanceWindowResponse;
import software.amazon.awssdk.services.ssm.model.CreateOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.CreateOpsItemResponse;
import software.amazon.awssdk.services.ssm.model.DeleteDocumentRequest;
import software.amazon.awssdk.services.ssm.model.DeleteMaintenanceWindowRequest;
import software.amazon.awssdk.services.ssm.model.DeleteOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.DescribeDocumentRequest;
import software.amazon.awssdk.services.ssm.model.DescribeDocumentResponse;
import software.amazon.awssdk.services.ssm.model.DescribeMaintenanceWindowsRequest;
import software.amazon.awssdk.services.ssm.model.DescribeMaintenanceWindowsResponse;
import software.amazon.awssdk.services.ssm.model.DescribeOpsItemsRequest;
import software.amazon.awssdk.services.ssm.model.DescribeOpsItemsResponse;
import software.amazon.awssdk.services.ssm.model.DocumentAlreadyExistsException;
import software.amazon.awssdk.services.ssm.model.DocumentType;
import software.amazon.awssdk.services.ssm.model.GetCommandInvocationRequest;
import software.amazon.awssdk.services.ssm.model.GetCommandInvocationResponse;
import software.amazon.awssdk.services.ssm.model.GetOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.GetOpsItemResponse;
import software.amazon.awssdk.services.ssm.model.ListCommandInvocationsRequest;
import software.amazon.awssdk.services.ssm.model.ListCommandInvocationsResponse;
import software.amazon.awssdk.services.ssm.model.MaintenanceWindowFilter;
import software.amazon.awssdk.services.ssm.model.MaintenanceWindowIdentity;
import software.amazon.awssdk.services.ssm.model.OpsItemDataValue;
import software.amazon.awssdk.services.ssm.model.OpsItemFilter;
import software.amazon.awssdk.services.ssm.model.OpsItemFilterKey;
import software.amazon.awssdk.services.ssm.model.OpsItemFilterOperator;
import software.amazon.awssdk.services.ssm.model.OpsItemStatus;
```

```
import software.amazon.awssdk.services.ssm.model.OpsItemSummary;
import software.amazon.awssdk.services.ssm.model.SendCommandRequest;
import software.amazon.awssdk.services.ssm.model.SendCommandResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;
import software.amazon.awssdk.services.ssm.model.UpdateMaintenanceWindowRequest;
import software.amazon.awssdk.services.ssm.model.UpdateOpsItemRequest;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html
 *
 * This Java program performs these tasks:
 * 1. Creates an AWS Systems Manager maintenance window with a default name or a
 * user-provided name.
 * 2. Modifies the maintenance window schedule.
 * 3. Creates a Systems Manager document with a default name or a user-provided
 * name.
 * 4. Sends a command to a specified EC2 instance using the created Systems Manager
 * document and displays the time when the command was invoked.
 * 5. Creates a Systems Manager OpsItem with a predefined title, source, category,
 * and severity.
 * 6. Updates and resolves the created OpsItem.
 * 7. Deletes the Systems Manager maintenance window, OpsItem, and document.
 */

public class SSMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws InterruptedException {
        String usage = ""
            Usage:
            <instanceId> <title> <source> <category> <severity>
    }
}
```

```
Where:
    instanceId - The Amazon EC2 Linux/UNIX instance Id that AWS Systems
Manager uses (ie, i-0149338494ed95f06).
    title - The title of the parameter (default is Disk Space Alert).
    source - The source of the parameter (default is EC2).
    category - The category of the parameter. Valid values are
'Availability', 'Cost', 'Performance', 'Recovery', 'Security' (default is
Performance).
    severity - The severity of the parameter. Severity should be a
number from 1 to 4 (default is 2).
    """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

Scanner scanner = new Scanner(System.in);
String documentName;
String windowName;
String instanceId = args[0];
String title = "Disk Space Alert" ;
String source = "EC2" ;
String category = "Performance" ;
String severity = "2" ;

Region region = Region.US_EAST_1;
SsmClient ssmClient = SsmClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("""
    Welcome to the AWS Systems Manager SDK Getting Started scenario.
    This program demonstrates how to interact with Systems Manager using the
AWS SDK for Java (v2).
    Systems Manager is the operations hub for your AWS applications and
resources and a secure end-to-end management solution.
    The program's primary functions include creating a maintenance window,
creating a document, sending a command to a document,
    listing documents, listing commands, creating an OpsItem, modifying an
OpsItem, and deleting Systems Manager resources.
    Upon completion of the program, all AWS resources are cleaned up.
    Let's get started...
    """);
```

```
        Please hit Enter
        """);
scanner.nextLine();
System.out.println(DASHES);

System.out.println("Create a Systems Manager maintenance window.");
System.out.println("Please enter the maintenance window name (default is
ssm-maintenance-window):");
String win = scanner.nextLine();
windowName = win.isEmpty() ? "ssm-maintenance-window" : win;
String winId = createMaintenanceWindow(ssmClient, windowName);
System.out.println(DASHES);

System.out.println("Modify the maintenance window by changing the
schedule");
System.out.println("Please hit Enter");
scanner.nextLine();
updateSSMMaintenanceWindow(ssmClient, winId, windowName);
System.out.println(DASHES);

System.out.println("Create a document that defines the actions that Systems
Manager performs on your EC2 instance.");
System.out.println("Please enter the document name (default is
ssmdocument):");
String doc = scanner.nextLine();
documentName = doc.isEmpty() ? "ssmdocument" : doc;
createSSMDoc(ssmClient, documentName);

System.out.println("Now we are going to run a command on an EC2 instance
that echoes 'Hello, world!'");
System.out.println("Please hit Enter");
scanner.nextLine();
String commandId = sendSSMCommand(ssmClient, documentName, instanceId);
System.out.println(DASHES);

System.out.println("Lets get the time when the specific command was sent to
the specific managed node");
System.out.println("Please hit Enter");
scanner.nextLine();
displayCommands(ssmClient, commandId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
```

Now we will create a `Systems Manager OpsItem`.  
An `OpsItem` is a feature provided by the `Systems Manager` service.  
It is a type of operational data item that allows you to manage and track various operational issues, events, or tasks within your AWS environment.

You can create `OpsItems` to track and manage operational issues as they arise.

For example, you could create an `OpsItem` whenever your application detects a critical error or an anomaly in your infrastructure.

```
""");

System.out.println("Please hit Enter");
scanner.nextLine();
String opsItemId = createSSMOpsItem(ssmClient, title, source, category,
severity);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Now we will update the OpsItem "+opsItemId);
System.out.println("Please hit Enter");
scanner.nextLine();
String description = "An update to "+opsItemId ;
updateOpsItem(ssmClient, opsItemId, title, description);
System.out.println("Now we will get the status of the OpsItem "+opsItemId);
System.out.println("Please hit Enter");
scanner.nextLine();
describeOpsItems(ssmClient, opsItemId);
System.out.println("Now we will resolve the OpsItem "+opsItemId);
System.out.println("Please hit Enter");
scanner.nextLine();
resolveOpsItem(ssmClient, opsItemId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Would you like to delete the Systems Manager resources?
(y/n)");
String delAns = scanner.nextLine().trim();
if (delAns.equalsIgnoreCase("y")) {
    System.out.println("You selected to delete the resources.");
    System.out.print("Press Enter to continue...");
    scanner.nextLine();
    deleteOpsItem(ssmClient, opsItemId);
```



```
        deleteMaintenanceWindow(ssmClient, winId);
        deleteDoc(ssmClient, documentName);
    } else {
        System.out.println("The Systems Manager resources will not be deleted");
    }
    System.out.println(DASHES);

    System.out.println("This concludes the Systems Manager SDK Getting Started
scenario.");
    System.out.println(DASHES);
}

// Displays the date and time when the specific command was invoked.
public static void displayCommands(SsmClient ssmClient, String commandId) {
    try {
        ListCommandInvocationsRequest commandInvocationsRequest =
ListCommandInvocationsRequest.builder()
            .commandId(commandId)
            .build();

        ListCommandInvocationsResponse response =
ssmClient.listCommandInvocations(commandInvocationsRequest);
        List<CommandInvocation> commandList = response.commandInvocations();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss").withZone(ZoneId.systemDefault());
        for (CommandInvocation invocation : commandList) {
            System.out.println("The time of the command invocation is " +
formatter.format(invocation.requestedDateTime()));
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Create an SSM OpsItem
public static String createSSMOpsItem(SsmClient ssmClient, String title, String
source, String category, String severity) {
    try {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
            .description("Created by the Systems Manager Java API")
            .title(title)
            .source(source)
```

```
        .category(category)
        .severity(severity)
        .build();

        CreateOpsItemResponse itemResponse =
ssmClient.createOpsItem(opsItemRequest);
        return itemResponse.opsItemId();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Update the AWS SSM OpsItem.
public static void updateOpsItem(SsmClient ssmClient, String opsItemId, String
title, String description) {
    Map<String, OpsItemDataValue> operationalData = new HashMap<>();
    operationalData.put("key1",
OpsItemDataValue.builder().value("value1").build());
    operationalData.put("key2",
OpsItemDataValue.builder().value("value2").build());

    try {
        UpdateOpsItemRequest request = UpdateOpsItemRequest.builder()
            .opsItemId(opsItemId)
            .title(title)
            .operationalData(operationalData)
            .status(getOpsItem(ssmClient, opsItemId))
            .description(description)
            .build();

        ssmClient.updateOpsItem(request);

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void resolveOpsItem(SsmClient ssmClient, String opsID) {
    try {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
```

```
        .opsItemId(opsID)
        .status(OpsItemStatus.RESOLVED)
        .build();

    ssmClient.updateOpsItem(opsItemRequest);

} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Gets a specific OpsItem.
private static OpsItemStatus getOpsItem(SsmClient ssmClient, String opsItemId) {
    GetOpsItemRequest itemRequest = GetOpsItemRequest.builder()
        .opsItemId(opsItemId)
        .build();

    try {
        GetOpsItemResponse response = ssmClient.getOpsItem(itemRequest);
        return response.opsItem().status();
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return null;
}

// Sends a SSM command to a managed node.
public static String sendSSMCommand(SsmClient ssmClient, String documentName,
String instanceId) throws InterruptedException {
    // Before we use Document to send a command - make sure it is active.
    boolean isDocumentActive = false;
    DescribeDocumentRequest request = DescribeDocumentRequest.builder()
        .name(documentName)
        .build();

    while (!isDocumentActive) {
        DescribeDocumentResponse response = ssmClient.describeDocument(request);
        String documentStatus = response.document().statusAsString();
        if (documentStatus.equals("Active")) {
            System.out.println("The Systems Manager document is active and ready
to use.");
        }
    }
}
```

```
        isDocumentActive = true;
    } else {
        System.out.println("The Systems Manager document is not active.
Status: " + documentStatus);
        try {
            // Add a delay to avoid making too many requests.
            Thread.sleep(5000); // Wait for 5 seconds before checking again
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

// Create the SendCommandRequest.
SendCommandRequest commandRequest = SendCommandRequest.builder()
    .documentName(documentName)
    .instanceIds(instanceId)
    .build();

// Send the command.
SendCommandResponse commandResponse = ssmClient.sendCommand(commandRequest);
String commandId = commandResponse.command().commandId();
System.out.println("The command Id is " + commandId);

// Wait for the command execution to complete.
GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
    .commandId(commandId)
    .instanceId(instanceId)
    .build();

System.out.println("Wait 5 secs");
TimeUnit.SECONDS.sleep(5);

// Retrieve the command execution details.
GetCommandInvocationResponse commandInvocationResponse =
ssmClient.getCommandInvocation(invocationRequest);

// Check the status of the command execution.
CommandInvocationStatus status = commandInvocationResponse.status();
if (status == CommandInvocationStatus.SUCCESS) {
    System.out.println("Command execution successful.");
} else {
    System.out.println("Command execution failed. Status: " + status);
}
```

```
    }
    return commandId;
}

// Deletes an AWS Systems Manager document.
public static void deleteDoc(SsmClient ssmClient, String documentName) {
    try {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
            .name(documentName)
            .build();

        ssmClient.deleteDocument(documentRequest);
        System.out.println("The Systems Manager document was successfully
deleted.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteMaintenanceWindow(SsmClient ssmClient, String winId) {
    try {
        DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
            .windowId(winId)
            .build();

        ssmClient.deleteMaintenanceWindow(windowRequest);
        System.out.println("The maintenance window was successfully deleted.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Update the maintenance window schedule
public static void updateSSMMaintenanceWindow(SsmClient ssmClient, String id,
String name) {
    try {
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
            .windowId(id)
```

```
        .allowUnassociatedTargets(true)
        .duration(24)
        .enabled(true)
        .name(name)
        .schedule("cron(0 0 ? * MON *)")
        .build();

    ssmClient.updateMaintenanceWindow(updateRequest);
    System.out.println("The Systems Manager maintenance window was
successfully updated.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String createMaintenanceWindow(SsmClient ssmClient, String
winName) {
    CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
        .name(winName)
        .description("This is my maintenance window")
        .allowUnassociatedTargets(true)
        .duration(2)
        .cutoff(1)
        .schedule("cron(0 10 ? * MON-FRI *)")
        .build();

    try {
        CreateMaintenanceWindowResponse response =
ssmClient.createMaintenanceWindow(request);
        String maintenanceWindowId = response.windowId();
        System.out.println("The maintenance window id is " +
maintenanceWindowId);
        return maintenanceWindowId;

    } catch (DocumentAlreadyExistsException e) {
        System.err.println("The maintenance window already exists. Moving on.");
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
    .key("name")
    .values(winName)
    .build();

DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
    .filters(filter)
    .build();

String windowId = "";
DescribeMaintenanceWindowsResponse response =
ssmClient.describeMaintenanceWindows(winRequest);
List<MaintenanceWindowIdentity> windows = response.windowIdentities();
if (!windows.isEmpty()) {
    windowId = windows.get(0).windowId();
    System.out.println("Window ID: " + windowId);
} else {
    System.out.println("Window not found.");
}
return windowId;
}

// Create an AWS SSM document to use in this scenario.
public static void createSSMDoc(SsmClient ssmClient, String docName) {
    // Create JSON for the content
    String jsonData = ""
    {
        "schemaVersion": "2.2",
        "description": "Run a simple shell command",
        "mainSteps": [
            {
                "action": "aws:runShellScript",
                "name": "runEchoCommand",
                "inputs": {
                    "runCommand": [
                        "echo 'Hello, world!'"
                    ]
                }
            }
        ]
    }
    "";
```

```
try {
    CreateDocumentRequest request = CreateDocumentRequest.builder()
        .content(jsonData)
        .name(docName)
        .documentType(DocumentType.COMMAND)
        .build();

    // Create the document.
    CreateDocumentResponse response = ssmClient.createDocument(request);
    System.out.println("The status of the document is " +
response.documentDescription().status());

} catch (DocumentAlreadyExistsException e) {
    System.err.println("The document already exists. Moving on." );
} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void describeOpsItems(SsmClient ssmClient, String key) {
    try {
        OpsItemFilter filter = OpsItemFilter.builder()
            .key(OpsItemFilterKey.OPS_ITEM_ID)
            .values(key)
            .operator(OpsItemFilterOperator.EQUAL)
            .build();

        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
            .maxResults(10)
            .opsItemFilters(filter)
            .build();

        DescribeOpsItemsResponse itemsResponse =
ssmClient.describeOpsItems(itemsRequest);
        List<OpsItemSummary> items = itemsResponse.opsItemSummaries();
        for (OpsItemSummary item : items) {
            System.out.println("The item title is " + item.title() +" and the
status is "+item.status().toString());
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```



```
    }  
  }  
  
  public static void deleteOpsItem(SsmClient ssmClient, String opsId) {  
    try {  
      DeleteOpsItemRequest deleteOpsItemRequest =  
DeleteOpsItemRequest.builder()  
        .opsItemId(opsId)  
        .build();  
  
      ssmClient.deleteOpsItem(deleteOpsItemRequest);  
      System.out.println(opsId + " Opsitem was deleted");  
  
    } catch (SsmException e) {  
      System.err.println(e.getMessage());  
      System.exit(1);  
    }  
  }  
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - [CommandInvocations](#)
  - [CreateDocument](#)
  - [CreateMaintenanceWindow](#)
  - [CreateOpsItem](#)
  - [DeleteMaintenanceWindow](#)
  - [SendCommand](#)
  - [UpdateOpsItem](#)

## SDK for Java 2.x を使用する Amazon Textract の例

次のコード例は、Amazon Textract AWS SDK for Java 2.x でを使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- [アクション](#)

## アクション

### AnalyzeDocument

次の例は、AnalyzeDocument を使用する方法を説明しています。

SDK for Java 2.x

#### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
}
}
```

- API の詳細については、「API リファレンス [AnalyzeDocument](#)」の「」を参照してください。  
AWS SDK for Java 2.x

## DetectDocumentText

次の例は、DetectDocumentText を使用する方法を説明しています。

## SDK for Java 2.x

**Note**

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

入ドキュメントからテキストを検出します。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectDocumentText {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <sourceDoc>.\s

                Where:
                sourceDoc - The path where the document is located (must be an
                image, for example, C:/AWS/book.png).\s
```

```
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String sourceDoc = args[0];
    Region region = Region.US_EAST_2;
    TextractClient textractClient = TextractClient.builder()
        .region(region)
        .build();

    detectDocText(textractClient, sourceDoc);
    textractClient.close();
}

public static void detectDocText(TextractClient textractClient, String
sourceDoc) {
    try {
        InputStream sourceStream = new FileInputStream(new File(sourceDoc));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Get the input Document object as bytes.
        Document myDoc = Document.builder()
            .bytes(sourceBytes)
            .build();

        DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
            .document(myDoc)
            .build();

        // Invoke the Detect operation.
        DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);
        List<Block> docInfo = textResponse.blocks();
        for (Block block : docInfo) {
            System.out.println("The block type is " +
block.blockType().toString());
        }

        DocumentMetadata documentMetadata = textResponse.documentMetadata();
```

```
        System.out.println("The number of pages in the document is " +
documentMetadata.pages());

    } catch (TextractException | FileNotFoundException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

## Amazon S3 バケット内のドキュメントからテキストを検出する

```
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);
    }
}
}
```

- API の詳細については、「API リファレンス [DetectDocumentText](#)」の「」を参照してください。AWS SDK for Java 2.x

## StartDocumentAnalysis

次の例は、StartDocumentAnalysis を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
}
}
```

- APIの詳細については、「APIリファレンス[StartDocumentAnalysis](#)」の「」を参照してください。AWS SDK for Java 2.x

## SDK for Java 2.x を使用した Amazon Transcribe の例

次のコード例は、Amazon Transcribe AWS SDK for Java 2.x で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- [アクション](#)
- [シナリオ](#)

アクション

## ListTranscriptionJobs

次の例は、ListTranscriptionJobs を使用する方法を説明しています。

SDK for Java 2.x

### Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public class ListTranscriptionJobs {
    public static void main(String[] args) {
        TranscribeClient transcribeClient = TranscribeClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listTranscriptionJobs(transcribeClient);
    }

    public static void listTranscriptionJobs(TranscribeClient transcribeClient)
    {
        ListTranscriptionJobsRequest listJobsRequest =
        ListTranscriptionJobsRequest.builder()
            .build();

        transcribeClient.listTranscriptionJobsPaginator(listJobsRequest).stream()
```

```
        .flatMap(response -> response.transcriptionJobSummaries().stream())
        .forEach(jobSummary -> {
            System.out.println("Job Name: " +
jobSummary.transcriptionJobName());
            System.out.println("Job Status: " +
jobSummary.transcriptionJobStatus());
            System.out.println("Output Location: " +
jobSummary.outputLocationType());
            // Add more information as needed

            // Retrieve additional details for the job if necessary
            GetTranscriptionJobResponse jobDetails =
transcribeClient.getTranscriptionJob(
                GetTranscriptionJobRequest.builder()
                    .transcriptionJobName(jobSummary.transcriptionJobName())
                    .build());

            // Display additional details
            System.out.println("Language Code: " +
jobDetails.transcriptionJob().languageCode());
            System.out.println("Media Format: " +
jobDetails.transcriptionJob().mediaFormat());
            // Add more details as needed

            System.out.println("-----");
        });
    }
}
```

- API の詳細については、「API リファレンス [ListTranscriptionJobs](#)」の「」を参照してください。AWS SDK for Java 2.x

## StartTranscriptionJob

次の例は、StartTranscriptionJob を使用する方法を説明しています。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws URISyntaxException, ExecutionException, InterruptedException,
        LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .region(REGION)
            .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromMic()),
    getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }
    }
}
```

```
        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
            .languageCode(LanguageCode.EN_US.toString())
            .mediaEncoding(MediaEncoding.PCM)
            .mediaSampleRateHertz(mediaSampleRateHertz)
            .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
            .onResponse(r -> {
                System.out.println("Received Initial response");
            })
            .onError(e -> {
                System.out.println(e.getMessage());
                StringWriter sw = new StringWriter();
                e.printStackTrace(new PrintWriter(sw));
                System.out.println("Error Occurred: " + sw.toString());
            })
            .onComplete(() -> {
                System.out.println("=== All records stream successfully ===");
            })
            .subscriber(event -> {
                List<Result> results = ((TranscriptEvent)
event).transcript().results();
                if (results.size() > 0) {
                    if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {
                        System.out.println(results.get(0).alternatives().get(0).transcript());
                    }
                }
            })
    }
}
```



```
        }
    })
    .build();
}

private InputStream getStreamFromFile(String audioFileName) {
    try {
        File inputFile = new
File(getClass().getClassLoader().getResource(audioFileName).getFile());
        InputStream audioStream = new FileInputStream(inputFile);
        return audioStream;
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}

private static class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;
    private static Subscription currentSubscription;

    private AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {

        if (this.currentSubscription == null) {
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        } else {
            this.currentSubscription.cancel();
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        }
        s.onSubscribe(currentSubscription);
    }
}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);
}
```

```
SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
    this.subscriber = s;
    this.inputStream = inputStream;
}

@Override
public void request(long n) {
    if (n <= 0) {
        subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
    }

    demand.getAndAdd(n);

    executor.submit(() -> {
        try {
            do {
                ByteBuffer audioBuffer = getNextEvent();
                if (audioBuffer.remaining() > 0) {
                    AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                    subscriber.onNext(audioEvent);
                } else {
                    subscriber.onComplete();
                    break;
                }
            } while (demand.decrementAndGet() > 0);
        } catch (Exception e) {
            subscriber.onError(e);
        }
    });
}

@Override
public void cancel() {
    executor.shutdown();
}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer = null;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

    int len = 0;
```

```
        try {
            len = inputStream.read(audioBytes);

            if (len <= 0) {
                audioBuffer = ByteBuffer.allocate(0);
            } else {
                audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
            }
        } catch (IOException e) {
            throw new UncheckedIOException(e);
        }

        return audioBuffer;
    }

    private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
        return AudioEvent.builder()
            .audioChunk(SdkBytes.fromByteBuffer(bb))
            .build();
    }
}
}
```

- API の詳細については、「API リファレンス [StartTranscriptionJob](#)」の「」を参照してください。AWS SDK for Java 2.x

## シナリオ

### 音声の文字起こしとジョブデータを取得する

次のコードサンプルは、以下の操作方法を示しています。

- Amazon Transcribe で文字起こしジョブを開始します。
- ジョブが完了するまで待ちます。
- 書き起こしが保存されている URI を取得します。

詳細については、「[Amazon Transcribe の開始方法](#)」を参照してください。

## SDK for Java 2.x

 Note

については、「」を参照してください GitHub。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

PCM ファイルを書き起こします。

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

コンピュータのマイクからのストリーミングオーディオを文字起こしします。

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws URISyntaxException, ExecutionException, InterruptedException,
        LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .region(REGION)
            .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromMic()),
    getResponseHandler());

        result.get();
        client.close();
    }
}
```



```
}

private static InputStream getStreamFromMic() throws LineUnavailableException {

    // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
    int sampleRate = 16000;
    AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
    DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

    if (!AudioSystem.isLineSupported(info)) {
        System.out.println("Line not supported");
        System.exit(0);
    }

    TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
    line.open(format);
    line.start();

    InputStream audioStream = new AudioInputStream(line);
    return audioStream;
}

private static AwsCredentialsProvider getCredentials() {
    return DefaultCredentialsProvider.create();
}

private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
    return StartStreamTranscriptionRequest.builder()
        .languageCode(LanguageCode.EN_US.toString())
        .mediaEncoding(MediaEncoding.PCM)
        .mediaSampleRateHertz(mediaSampleRateHertz)
        .build();
}

private static StartStreamTranscriptionResponseHandler getResponseHandler() {
    return StartStreamTranscriptionResponseHandler.builder()
        .onResponse(r -> {
            System.out.println("Received Initial response");
        })
        .onError(e -> {
            System.out.println(e.getMessage());
            StringWriter sw = new StringWriter();
            e.printStackTrace(new PrintWriter(sw));
        });
}
```

```
        System.out.println("Error Occurred: " + sw.toString());
    })
    .onComplete(() -> {
        System.out.println("=== All records stream successfully ===");
    })
    .subscriber(event -> {
        List<Result> results = ((TranscriptEvent)
event).transcript().results();
        if (results.size() > 0) {
            if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

System.out.println(results.get(0).alternatives().get(0).transcript());
                }
            }
        })
    .build();
}

private InputStream getStreamFromFile(String audioFileName) {
    try {
        File inputFile = new
File(getClass().getClassLoader().getResource(audioFileName).getFile());
        InputStream audioStream = new FileInputStream(inputFile);
        return audioStream;
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}

private static class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;
    private static Subscription currentSubscription;

    private AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {

        if (this.currentSubscription == null) {
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        } else {
```

```
        this.currentSubscription.cancel();
        this.currentSubscription = new SubscriptionImpl(s, inputStream);
    }
    s.onSubscribe(currentSubscription);
}
}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);

    SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
    {
        this.subscriber = s;
        this.inputStream = inputStream;
    }

    @Override
    public void request(long n) {
        if (n <= 0) {
            subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
        }

        demand.getAndAdd(n);

        executor.submit(() -> {
            try {
                do {
                    ByteBuffer audioBuffer = getNextEvent();
                    if (audioBuffer.remaining() > 0) {
                        AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                        subscriber.onNext(audioEvent);
                    } else {
                        subscriber.onComplete();
                        break;
                    }
                } while (demand.decrementAndGet() > 0);
            } catch (Exception e) {
                subscriber.onError(e);
            }
        });
    }
}
```

```
        }
    });
}

@Override
public void cancel() {
    executor.shutdown();
}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer = null;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

    int len = 0;
    try {
        len = inputStream.read(audioBytes);

        if (len <= 0) {
            audioBuffer = ByteBuffer.allocate(0);
        } else {
            audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
        }
    } catch (IOException e) {
        throw new UncheckedIOException(e);
    }

    return audioBuffer;
}

private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [GetTranscriptionJob](#)
  - [StartTranscriptionJob](#)

## SDK for Java 2.x を使用したクロスサービスの例

以下のサンプルアプリケーションでは、AWS SDK for Java 2.x を使用して複数の AWS のサービスアプリケーションで動作します。

クロスサービスの例は、アプリケーションの構築を始めるのに役立つ上級レベルの経験を対象としています。

### 例

- [DynamoDB テーブルにデータを送信するアプリケーションを構築する](#)
- [Amazon Lex chatbot を作成して、ウェブサイトの訪問者に対応する](#)
- [メッセージを翻訳する公開およびサブスクリプションアプリケーションを構築する](#)
- [Amazon SQS を使用してメッセージを送受信するウェブアプリケーションを作成する](#)
- [ユーザーがラベルを使用して写真を管理できる写真アセット管理アプリケーションの作成](#)
- [DynamoDB データを追跡するウェブアプリケーションを作成する](#)
- [Amazon Redshift アイテムトラッカーの作成](#)
- [Aurora Serverless 作業項目トラッカーの作成](#)
- [顧客からのフィードバックを分析し、音声を合成するアプリケーションの作成](#)
- [AWS SDK を使用して Amazon Rekognition でイメージ内の PPE を検出する](#)
- [AWS SDK を使用して Amazon Rekognition でイメージ内のオブジェクトを検出する](#)
- [AWS SDK を使用して Amazon Rekognition でビデオ内のユーザーとオブジェクトを検出する](#)
- [AWS SDK を使用して Amazon DynamoDB のパフォーマンスをモニタリングする](#)
- [AWS SDK を使用して Amazon SNS メッセージを Amazon SQS キューに発行する](#)
- [API Gateway を使用して Lambda 関数を呼び出す](#)
- [Step Functions を使用して Lambda 関数を呼び出す](#)
- [スケジュールされたイベントを使用した Lambda 関数の呼び出し](#)

## DynamoDB テーブルにデータを送信するアプリケーションを構築する

### SDK for Java 2.x

Amazon DynamoDB Java API を使用してデータを送信し、Amazon Simple Notification Service Java API を使用してテキストメッセージを送信する動的ウェブアプリケーションを作成する方法について説明します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください [GitHub](#)。

この例で使用されているサービス

- DynamoDB
- Amazon SNS

## Amazon Lex chatbot を作成して、ウェブサイトの訪問者に対応する

SDK for Java 2.x

ウェブアプリケーション内に Amazon Lex chatbot を作成して、ウェブサイトの訪問者に対応することができます。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください [GitHub](#)。

この例で使用されているサービス

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

## メッセージを翻訳する公開およびサブスクリプションアプリケーションを構築する

SDK for Java 2.x

Amazon Simple Notification Service Java API を使用して、サブスクリプションおよびパブリッシュ機能を持つウェブアプリケーションを作成する方法を説明します。さらに、このサンプルアプリケーションではメッセージを翻訳します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください [GitHub](#)。

完全なソースコードと、Java 非同期 API を使用する例をセットアップして実行する方法については、の完全な例を参照してください [GitHub](#)。

この例で使用されているサービス

- Amazon SNS
- Amazon Translate

## Amazon SQS を使用してメッセージを送受信するウェブアプリケーションを作成する

SDK for Java 2.x

Amazon SQS API を使用して、メッセージを送受信する Spring REST API を開発する方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください [GitHub](#)。

この例で使用されているサービス

- Amazon Comprehend
- Amazon SQS

## ユーザーがラベルを使用して写真を管理できる写真アセット管理アプリケーションの作成

SDK for Java 2.x

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください [GitHub](#)。

この例のソースについては詳しくは、[AWS コミュニティ](#)でブログ投稿を参照してください。

この例で使用されているサービス

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition

- Amazon S3
- Amazon SNS

## DynamoDB データを追跡するウェブアプリケーションを作成する

### SDK for Java 2.x

Amazon DynamoDB API を使用して、DynamoDB 作業データを追跡する動的ウェブアプリケーションを作成する方法を示しています。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください [GitHub](#)。

この例で使用されているサービス

- DynamoDB
- Amazon SES

## Amazon Redshift アイテムトラッカーの作成

### SDK for Java 2.x

Amazon Redshift データベースに保存されている作業項目を追跡してレポートするウェブアプリケーションを作成する方法を説明します。

Amazon Redshift データをクエリし、React アプリケーションで使用する Spring REST API を設定する方法の完全なソースコードと手順については、「」の詳細な例を参照してください [GitHub](#)。

この例で使用されているサービス

- Amazon Redshift
- Amazon SES

## Aurora Serverless 作業項目トラッカーの作成

### SDK for Java 2.x

Amazon RDS データベースに保存されている作業項目を追跡およびレポートするウェブアプリケーションを作成する方法を説明します。



Amazon Aurora Serverless データをクエリし、React アプリケーションで使用する Spring REST API を設定する方法の完全なソースコードと手順については、「」の詳細な例を参照してください [GitHub](#)。

JDBC API を使用する例をセットアップして実行する方法の完全なソースコードと手順については、「」の詳細な例を参照してください [GitHub](#)。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS データサービス
- Amazon SES

## 顧客からのフィードバックを分析し、音声を合成するアプリケーションの作成

SDK for Java 2.x

このサンプルアプリケーションは、顧客フィードバックカードを分析し、保存します。具体的には、ニューヨーク市の架空のホテルのニーズを満たします。このホテルでは、お客様からのフィードバックをさまざまな言語で書かれた実際のコメントカードの形で受け取ります。そのフィードバックは、ウェブクライアントを通じてアプリにアップロードされます。コメントカードの画像をアップロードされると、次の手順が発生します。

- テキストは Amazon Textract を使用して、画像から抽出されます。
- Amazon Comprehend は、抽出されたテキストの感情とその言語を決定します。
- 抽出されたテキストは、Amazon Translate を使用して英語に翻訳されます。
- Amazon Polly は抽出されたテキストからオーディオファイルを合成します。

完全なアプリは AWS CDK を使用してデプロイすることができます。ソースコードとデプロイ手順については、「」の「[GitHub](#)プロジェクト」を参照してください。

この例で使用されているサービス

- Amazon Comprehend
- Lambda

- Amazon Polly
- Amazon Textract
- Amazon Translate

## AWS SDK を使用して Amazon Rekognition でイメージ内の PPE を検出する

### SDK for Java 2.x

個人用保護具を使用して画像を検出する AWS Lambda 関数を作成する方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください [GitHub](#)。

この例で使用されているサービス

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

## AWS SDK を使用して Amazon Rekognition でイメージ内のオブジェクトを検出する

### SDK for Java 2.x

Amazon Rekognition Java API を使用して、Amazon Rekognition を使用し、 Amazon Simple Storage Service (Amazon S3) バケットにあるイメージ内でカテゴリ別にオブジェクトを識別するアプリケーションを作成する方法について説明します。アプリケーションは Amazon Simple Email Service (Amazon SES) を使用して、結果を含む E メール通知を管理者に送信します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください [GitHub](#)。

この例で使用されているサービス

- Amazon Rekognition

- Amazon S3
- Amazon SES

## AWS SDK を使用して Amazon Rekognition でビデオ内のユーザーとオブジェクトを検出する

### SDK for Java 2.x

Amazon Rekognition Java API を使用し、Amazon Simple Storage Service (Amazon S3) バケツトにあるビデオ内で顔やオブジェクトを検出するアプリケーションを作成する方法について説明します。アプリケーションは Amazon Simple Email Service (Amazon SES) を使用して、結果を含む E メール通知を管理者に送信します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください [GitHub](#)。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES

## AWS SDK を使用して Amazon DynamoDB のパフォーマンスをモニタリングする

### SDK for Java 2.x

この例では、DynamoDB のパフォーマンスをモニタリングするように Java アプリケーションを設定する方法を示します。アプリケーションは、パフォーマンスをモニタリング CloudWatch できる にメトリクスデータを送信します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください [GitHub](#)。

この例で使用されているサービス

- CloudWatch

- DynamoDB

## AWS SDK を使用して Amazon SNS メッセージを Amazon SQS キューに発行する

### SDK for Java 2.x

Amazon Simple Notification Service (Amazon SNS) と Amazon Simple Queue Service (Amazon SQS) でのトピックとキューを使用したメッセージングを示します。

Amazon SNS および Amazon SQS のトピックとキューを使用したメッセージングの完全なソースコードと手順については、「」の詳細な例を参照してください[GitHub](#)。

この例で使用されているサービス

- Amazon SNS
- Amazon SQS

## API Gateway を使用して Lambda 関数を呼び出す

### SDK for Java 2.x

Lambda Java ランタイム API を使用して AWS Lambda 関数を作成する方法を示します。この例では、さまざまな AWS サービスを呼び出して、特定のユースケースを実行します。この例では、Amazon API Gateway によって呼び出される Lambda 関数を作成する方法を示します。この関数は、Amazon DynamoDB テーブルをスキャンして、Amazon Simple Notification Service (Amazon SNS) を使用して、従業員に年間の記念日を祝福するテキストメッセージを送信します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください[GitHub](#)。

この例で使用されているサービス

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

## Step Functions を使用して Lambda 関数を呼び出す

### SDK for Java 2.x

AWS Step Functions とを使用して AWS サーバーレスワークフローを作成する方法を示します。AWS SDK for Java 2.x。各ワークフローステップは、AWS Lambda 関数を使用して実装されません。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください [GitHub](#)。

この例で使用されているサービス

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

## スケジュールされたイベントを使用した Lambda 関数の呼び出し

### SDK for Java 2.x

AWS Lambda 関数を呼び出す Amazon EventBridge スケジュールされたイベントを作成する方法を示します。cron 式 EventBridge を使用して Lambda 関数が呼び出されるタイミングをスケジュールするようにを設定します。この例では、Lambda Java ランタイム API を使用して Lambda 関数を作成します。この例では、さまざまな AWS サービスを呼び出して、特定のユースケースを実行します。この例では、年間の記念日に従業員を祝福するモバイルテキストメッセージを従業員に送信するアプリを作成する方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください [GitHub](#)。

この例で使用されているサービス

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

# のセキュリティ AWS SDK for Java

クラウドセキュリティは Amazon Web Services (AWS) の最優先事項です。AWS のお客様は、セキュリティを非常に重視する組織の要件を満たせるように構築されたデータセンターとネットワークアーキテクチャーから利点を得ます。セキュリティは、AWS とユーザーの間で共有される責任です。[責任共有モデル](#)では、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

クラウドのセキュリティ — AWS クラウドで提供されるすべてのサービスを実行するインフラストラクチャ AWS を保護し、安全に使用できるサービスを提供します。当社のセキュリティ責任は、最優先事項であり AWS、当社のセキュリティの有効性は、[AWS コンプライアンスプログラムの一環として、サードパーティーの監査者によって定期的にテストおよび検証されています](#)。

クラウドにおけるセキュリティ — お客様の責任は、使用している AWS サービス、およびデータの機密性、組織の要件、適用される法律や規制などのその他の要因によって決まります。

この AWS 製品またはサービスは、サポートする特定の Amazon Web Services (AWS) サービスを通じて[責任共有モデル](#)に従います。AWS サービスセキュリティ情報については、[AWS 「サービスセキュリティドキュメント」ページ](#)と[AWS、AWS コンプライアンスプログラムによるコンプライアンスの取り組みの対象となるサービス](#)を参照してください。

## トピック

- [AWS SDK for Java 2.x でのデータ保護](#)
- [SDK for Java で TLS の使用](#)
- [Identity and Access Management](#)
- [この AWS 製品またはサービスのコンプライアンス検証](#)
- [この AWS 製品またはサービスの耐障害性](#)
- [この AWS 製品またはサービスのインフラストラクチャセキュリティ](#)

## AWS SDK for Java 2.x でのデータ保護

責任 [AWS 共有モデル](#)、でのデータ保護に適用されます AWS SDK for Java。このモデルで説明されているように、AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。お客様は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。また、使用する AWS のサービスのセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「[データプライバシーの](#)

[よくある質問](#)」を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された記事「[AWS 責任共有モデルおよび GDPR](#)」を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント、AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須であり TLS 1.3 がお勧めです。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。
- AWS 暗号化ソリューションと、内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「[連邦情報処理規格 \(FIPS\) 140-2](#)」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、コンソール、API、または SDK を使用して AWS CLI SDK for Java または他の AWS のサービス AWS SDKs。名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

## SDK for Java で TLS の使用

は、基盤となる Java プラットフォームの TLS 機能 AWS SDK for Java を使用します。このトピックでは、[Amazon Corretto 17](#) で使用されている OpenJDK 実装を使用した例を紹介します。

を使用するには AWS のサービス、基盤となる JDK が TLS 1.2 の最小バージョンをサポートしている必要がありますが、TLS 1.3 が推奨されます。

ユーザーは、SDK で使用している Java プラットフォームのドキュメントを参照して、デフォルトで有効になっている TLS バージョンと、特定の TLS バージョンを有効または無効にする方法を確認する必要があります。

## TLS バージョン情報を確認する方法

OpenJDK を使用した次のコードは、[SSLContext](#) を使用して、サポートされている TLS/SSL バージョンを出力する方法を示しています。

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProtocols()));
```

たとえば、Amazon Corretto 17 (OpenJDK) は以下の出力を生成します。

```
[TLSv1.3, TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2Hello]
```

動作中の SSL ハンドシェイクと使用されている TLS のバージョンを確認するには、システムプロパティ `javax.net.debug` を使用します。

たとえば、TLS を使用する Java アプリケーションを実行します。

```
java app.jar -Djavax.net.debug=ssl:handshake
```

アプリケーションでは、次のような SSL ハンドシェイクがログ記録されます。

```
...
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",
  ...
}
...
```

## 最小 TLS バージョンを強制する

SDK for Java は、プラットフォームとサービスでサポートされている最新の TLS バージョンを常に優先します。特定の最小 TLS バージョンを強制する場合は、Java プラットフォームのドキュメントを確認してください。

OpenJDK ベースの JVM では、システムプロパティ `jdk.tls.client.protocols` を使用できません。



たとえば、TLS 1.3 が利用可能であっても、アプリケーション内の SDK サービスクライアントに TLS 1.2 を使用させたい場合は、次のシステムプロパティを指定します。

```
java app.jar -Djdk.tls.client.protocols=TLSv1.2
```

## AWS API エンドポイントの TLS 1.2 へのアップグレード

最小バージョンの TLS 1.2 に移行する AWS API エンドポイントの詳細については、この[ブログ記事](#)を参照してください。

## Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つです。IAM 管理者は、誰を認証 (サインイン) し、誰に AWS リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで AWS のサービス 使用できる です。

### トピック

- [対象者](#)
- [アイデンティティを使用した認証](#)
- [ポリシーを使用したアクセスの管理](#)
- [IAM の AWS のサービス 仕組み](#)
- [AWS ID とアクセスのトラブルシューティング](#)

### 対象者

AWS Identity and Access Management (IAM) の使用方法は、 で行う作業によって異なります AWS。

サービスユーザー – AWS のサービス を使用してジョブを実行する場合、管理者から必要な認証情報とアクセス許可が与えられます。さらに多くの AWS 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解しておく、管理者に適切な許可をリクエストするうえで役立ちます。の機能にアクセスできない場合は AWS、[AWS ID とアクセスのトラブルシューティング](#)「」または AWS のサービス 使用している のユーザーガイドを参照してください。

サービス管理者 – 社内の AWS リソースを担当している場合は、通常、へのフルアクセスがあります AWS。サービスユーザーがどの AWS 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社で IAM をで使用する方法的詳細については AWS、使用している の AWS のサービス ユーザーガイドを参照してください。

IAM 管理者 - 管理者は、AWSへのアクセスを管理するポリシーの書き込み方法の詳細について確認する場合があります。IAM で使用できる AWS アイデンティティベースのポリシーの例を表示するには、AWS のサービス 使用している のユーザーガイドを参照してください。

## アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けて認証 ( にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーテッド ID AWS として にサインインできます。AWS IAM Identity Center ( IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーテッド ID の例です。フェデレーテッドアイデンティティとしてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「ユーザーガイド」の「[にサインインする方法 AWS アカウント](#)」を参照してください。

AWS プログラムで にアクセスする場合、 は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストに暗号で署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。推奨される方法を使用してリクエストを自分で署名する方法の詳細については、IAM [ユーザーガイドの API AWS リクエスト](#) の署名を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、AWS では、多要素認証 (MFA) を使用してアカウントのセキュリティを向上させることをお勧めします。詳細については、『AWS IAM Identity Center ユーザーガイド』の「[Multi-factor authentication](#)」(多要素認証) および『IAM ユーザーガイド』の「[AWSにおける多要素認証 \(MFA\) の使用](#)」を参照してください。

## AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての およびリソースへの AWS のサービス 完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウ ント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサイン インすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強く お勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実 行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストに ついては、『IAM ユーザーガイド』の「[ルートユーザー認証情報が必要なタスク](#)」を参照してくだ さい。

## フェデレーティッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに、一時 的な認証情報を使用して にアクセスするための ID プロバイダーとのフェデレーションの使用を要求 AWS のサービスします。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、 AWS Directory Service、アイデンティティセンターディレクトリのユーザー、または ID ソースを 通じて提供された認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレー ティッド ID が にアクセスすると AWS アカウント、ロールが引き受けられ、ロールは一時的な認証 情報を提供します。

アクセスを一元管理する場合は、AWS IAM Identity Centerを使用することをお勧めします。IAM Identity Center でユーザーとグループを作成することも、独自の ID ソース内のユーザーとグルー プのセットに接続して同期して、すべての AWS アカウント とアプリケーションで使用することも できます。IAM Identity Center の詳細については、『AWS IAM Identity Center ユーザーガイド』の 「[What is IAM Identity Center?](#)」(IAM Identity Center とは) を参照してください。

## IAM ユーザーとグループ

[IAM ユーザー](#)は、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウ ント を持つ 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な 認証情報を保有する IAM ユーザーを作成する代わりに、一時認証情報を使用することをお勧めしま す。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アク セスキーをローテーションすることをお勧めします。詳細については、IAM ユーザーガイドの「[長 期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする](#)」を 参照してください。

[IAM グループ](#)は、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する権限を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、『IAM ユーザーガイド』の「[IAM ユーザー \(ロールではなく\) の作成が適している場合](#)」を参照してください。

## IAM ロール

[IAM ロール](#)は、特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロール を切り替える AWS Management Console ことで、[で IAM ロール](#)を一時的に引き受けることができます。ロールを引き受けるには、または AWS API AWS CLI オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「[IAM ロールの使用](#)」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス - フェデレーテッドアイデンティティに権限を割り当てるには、ロールを作成してそのロールの権限を定義します。フェデレーテッドアイデンティティが認証されると、そのアイデンティティはロールに関連付けられ、ロールで定義されている権限が付与されます。フェデレーションの詳細については、『IAM ユーザーガイド』の「[サードパーティーアイデンティティプロバイダー向けロールの作成](#)」を参照してください。IAM アイデンティティセンターを使用する場合、権限セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。権限セットの詳細については、『AWS IAM Identity Center ユーザーガイド』の「[権限セット](#)」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、(ロールをプロキシとして使用する代わりに) ポリシーをリソースに直接アタッチできま

- す。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、『IAM ユーザーガイド』の「[IAM ロールとリソースベースのポリシーとの相違点](#)」を参照してください。
- クロスサービスアクセス — 一部の は、他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの権限、サービスロール、またはサービスにリンクされたロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して でアクションを実行する場合 AWS、ユーザーはプリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストのリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「[転送アクセスセッション](#)」を参照してください。
  - サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける [IAM ロール](#)です。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、IAM ユーザーガイドの「[AWS のサービスにアクセス許可を委任するロールの作成](#)」を参照してください。
  - サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの権限を表示できますが、編集することはできません。
  - Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、『IAM ユーザーガイド』の「[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して権限を付与する](#)」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、『IAM ユーザーガイド』の「[IAM ユーザーではなく\) IAM ロールをいつ作成したら良いのか?](#)」を参照してください。

## ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、AWS ID またはリソースにアタッチします。ポリシーは、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義するオブジェクトです。は、プリンシパル(ユーザー、ルートユーザー、またはロールセッション) AWS がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュメント AWS としてに保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「[JSON ポリシー概要](#)」を参照してください。

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの権限を定義します。例えば、iam:GetRole アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、AWS Management Console、AWS CLI または AWS API からロール情報を取得できます。

### アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、IAM ユーザーガイドの「[IAM ポリシーの作成](#)」を参照してください。

アイデンティティベースポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、『IAM ユーザーガイド』の「[マネージドポリシーとインラインポリシーの比較](#)」を参照してください。

## リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、[プリンシパルを指定する](#)必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、またはを含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

## アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための権限を持つかをコントロールします。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、AWS WAF、および Amazon VPC は、ACLs。ACL の詳細については、『Amazon Simple Storage Service デベロッパーガイド』の「[アクセスコントロールリスト \(ACL\) の概要](#)」を参照してください。

## その他のポリシータイプ

AWS は、一般的ではない追加のポリシータイプをサポートします。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- **アクセス許可の境界** - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。Principal フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。許可の境界の詳細については、「IAM ユーザーガイド」の「[IAM エンティティの許可の境界](#)」を参照してください。
- **サービスコントロールポリシー (SCPs)** – SCPs は、 の組織または組織単位 (OU) に対する最大アクセス許可を指定する JSON ポリシーです AWS Organizations。AWS Organizations は、AWS

アカウント ビジネスが所有する複数の をグループ化して一元管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、『AWS Organizations ユーザーガイド』の「[SCP の仕組み](#)」を参照してください。

- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合があります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、『IAM ユーザーガイド』の「[セッションポリシー](#)」を参照してください。

## 複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、IAM ユーザーガイドの「[ポリシー評価ロジック](#)」を参照してください。

## IAM の AWS のサービス 仕組み

ほとんどの IAM 機能との AWS のサービス 連携方法の概要については、IAM ユーザーガイドの[AWS 「IAM と連携する のサービス」](#)を参照してください。

IAM AWS のサービス で特定の を使用する方法については、関連するサービスのユーザーガイドのセキュリティセクションを参照してください。

## AWS ID とアクセスのトラブルシューティング

次の情報は、 と IAM の使用時に発生する可能性がある一般的な問題の診断 AWS と修正に役立ちます。

### トピック

- [でアクションを実行する権限がない AWS](#)
- [iam を実行する権限がありません。PassRole](#)
- [自分の 以外のユーザーに自分の AWS リソース AWS アカウント へのアクセスを許可したい](#)



## でアクションを実行する権限がない AWS

「I am not authorized to perform an action in Amazon Bedrock」というエラーが表示された場合、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要な `aws:GetWidget` アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
aws:GetWidget on resource: my-example-widget
```

この場合、`aws:GetWidget` アクションを使用して *my-example-widget* リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

## iam を実行する権限がありません。PassRole

`iam:PassRole` アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して AWS にロールを渡すことができるようにする必要があります。

一部の AWS のサービスでは、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

以下の例のエラーは、marymajor という IAM ユーザーがコンソールを使用して AWS でアクションを実行しようする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。Mary には、ロールをサービスに渡す権限がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに `iam:PassRole` アクションの実行を許可する必要があります。

サポートが必要な場合は、AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分の 以外のユーザーに自分の AWS リソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- がこれらの機能 AWS をサポートしているかどうかを確認するには、「」を参照してください [IAM の AWS のサービス 仕組み](#)。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、[IAM ユーザーガイドの「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」](#)を参照してください。
- リソースへのアクセスをサードパーティー に提供する方法については AWS アカウント、IAM ユーザーガイドの [「サードパーティー AWS アカウント が所有する へのアクセスを提供する」](#)を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、『IAM ユーザーガイド』の [「外部で認証されたユーザー \(ID フェデレーション\) へのアクセス権限」](#)を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いの詳細については、「IAM ユーザーガイド」の [「IAM ロールとリソースベースのポリシーとの相違点」](#)を参照してください。


## この AWS 製品またはサービスのコンプライアンス検証

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、コンプライアンスプログラム [AWS のサービス による対象範囲内のコンプライアンスプログラム](#)を参照し、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、[AWS 「コンプライアンスプログラム」](#)を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「[でのレポートのダウンロード AWS Artifact](#)」の」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。では、コンプライアンスに役立つ以下のリソース AWS を提供しています。

- [セキュリティとコンプライアンスのクイックスタートガイド](#) – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境 AWS を にデプロイする手順について説明します。
- [アマゾン ウェブ サービスにおける HIPAA セキュリティとコンプライアンスのアーキテクチャ](#) – このホワイトペーパーでは、企業が AWS を使用して HIPAA 対象アプリケーションを作成する方法について説明します。

 Note

すべて AWS のサービス HIPAA の対象となるわけではありません。詳細については、「[HIPAA 対応サービスのリファレンス](#)」を参照してください。

- [AWS コンプライアンスリソース](#) – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- [AWS カスタマーコンプライアンスガイド](#) – コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、ガイダンスを保護し AWS のサービス、複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) を含む) のセキュリティコントロールにマッピングするためのベストプラクティスをまとめています。
- 「[デベロッパーガイド](#)」の「[ルールによるリソースの評価](#)」 – この AWS Config サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。AWS Config
- [AWS Security Hub](#) – これにより AWS のサービス、内のセキュリティ状態を包括的に確認できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、「[Security Hub のコントロールリファレンス](#)」を参照してください。
- [Amazon GuardDuty](#) – これにより AWS アカウント、疑わしいアクティビティや悪意のあるアクティビティがないか環境を監視することで、、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS のサービス を検出します。GuardDuty は、特定のコンプライアンスフレームワークで義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件への対応に役立ちます。
- [AWS Audit Manager](#) – これにより AWS のサービス、AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

この AWS 製品またはサービスは、サポートする特定の Amazon Web Services (AWS) サービスを通じて [責任共有モデル](#) に従います。AWS サービスセキュリティ情報については、[AWS 「サービスセキュリティドキュメント」 ページ](#) と [AWS 、 AWS コンプライアンスプログラム によるコンプライアンスの取り組みの対象となるサービス](#) を参照してください。

## この AWS 製品またはサービスの耐障害性

AWS グローバルインフラストラクチャは、AWS リージョン およびアベイラビリティゾーンを中心に構築されています。

AWS リージョン は、低レイテンシー、高スループット、高冗長ネットワークで接続された、物理的に分離および隔離された複数のアベイラビリティゾーンを提供します。

アベイラビリティゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性が高く、フォールトトレラントで、スケーラブルです。

AWS リージョンとアベイラビリティゾーンの詳細については、[AWS 「グローバルインフラストラクチャ」](#) を参照してください。

この AWS 製品またはサービスは、サポートする特定の Amazon Web Services (AWS) サービスを通じて [責任共有モデル](#) に従います。AWS サービスセキュリティ情報については、[AWS 「サービスセキュリティドキュメント」 ページ](#) と [AWS 、 AWS コンプライアンスプログラム によるコンプライアンスの取り組みの対象となるサービス](#) を参照してください。

## この AWS 製品またはサービスのインフラストラクチャセキュリティ

この AWS 製品またはサービスはマネージドサービスを使用するため、グローバルネットワークセキュリティによって保護されています AWS 。AWS セキュリティサービスと [インフラストラクチャ AWS を保護する方法](#) については、[AWS 「クラウドセキュリティ」](#) を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「Security Pillar AWS Well-Architected Framework」の [「Infrastructure Protection」](#) を参照してください。

が AWS 公開した API コールを使用して、ネットワーク経由でこの AWS 製品またはサービスにアクセスします。クライアントは以下をサポートする必要があります:

- Transport Layer Security (TLS)。TLS 1.2 は必須で TLS 1.3 がお勧めです。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは、Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、[AWS Security Token Service](#) (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

この AWS 製品またはサービスは、サポートする特定の Amazon Web Services (AWS) サービスを通じて [責任共有モデル](#) に従います。AWS サービスセキュリティ情報については、[AWS 「サービスセキュリティドキュメント」ページ](#) と [AWS、AWS コンプライアンスプログラムによるコンプライアンスの取り組みの対象となるサービス](#) を参照してください。

# のバージョン 1.x から 2.x への移行 AWS SDK for Java

AWS SDK for Java 2.x は、Java 8+ 上に構築された 1.x コードベースの主な書き換えです。これには、整合性と使いやすさの向上や強さに改善されたイミュータブル性などの多くの更新が含まれています。このセクションでは、バージョン 2.x の主な新機能の説明、およびバージョン 1.x から 2.x にコードを移行する方法についてのガイダンスが用意されています。

## トピック

- [バージョン 2 の新機能](#)
- [移行 step-by-step 手順と例](#)
- [パッケージ名から Maven artifactId へのマッピング](#)
- [AWS SDK for Java 1.x と 2.x の違い](#)
- [SDK for Java 1.x と 2.x を並行して使用する](#)

## バージョン 2 の新機能

- 独自の HTTP クライアントを設定できます。「[HTTP transport configuration](#)」を参照してください。
- 非同期クライアントは、ノンブロッキング I/O サポートを提供し、CompletableFuture オブジェクトを返します。[非同期プログラミング](#)を参照してください。
- 複数ページを返すオペレーションには、自動ページ分割レスポンスがあります。これにより、後続のページを確認して取得する必要がなくなり、レスポンスをどのように処理するかのコードに集中できます。「[ページ分割](#)」を参照してください。
- AWS Lambda 関数の SDK 開始時刻のパフォーマンスが向上しました。「[SDK start time performance improvements](#)」を参照してください。
- バージョン 2.x は、リクエストを作成するための、新しい簡易的な方法をサポートします。

### Example

```
dynamoDbClient.putItem(request -> request.tableName(TABLE))
```

新機能の詳細と特定のコード例については、このガイドの他のセクションを参照してください。

- [クイックスタート](#)

- [セットアップ](#)
- [AWS SDK for Java 2.x のコード例](#)
- [SDK を使用する](#)
- [のセキュリティ AWS SDK for Java](#)

## 移行 step-by-step 手順と例

このセクションでは、現在 SDK for Java v1.x を使用しているアプリケーションを SDK for Java 2.x に移行するための step-by-step ガイドを提供します。最初の部分では、ステップの概要を示し、その後に移行の詳細な例を示します。

ここで説明するステップでは、アプリケーションがモデル駆動型サービスクライアント AWS のサービスを使用して呼び出す通常のユースケースの移行について説明します。[S3 Transfer Manager](#) や [CloudFrontの事前署名](#) など、上位レベルの APIs を使用するコードを移行する必要がある場合は、[the section called “1.x と 2.x の相違点”](#) 目次の下のセクションを参照してください。

ここで説明するアプローチは提案です。他の手法を使用し、IDE のコード編集機能を活用して同じ結果を得ることができます。

### 手順の概要

#### 1. SDK for Java 2.x BOM を追加して開始する

SDK for Java 2.x の Maven BOM (部品表) 要素を POM ファイルに追加することで、必要なすべての v2 依存関係が同じバージョンであることを確認します。POM には、v1 と v2 の両方の依存関係を含めることができます。これにより、コードを一度にすべて変更するのではなく、段階的に移行できます。

#### SDK for Java 2.x BOM

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.24.3</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
```

```
</dependencies>  
</dependencyManagement>
```

[最新バージョン](#)は Maven Central Repository にあります。

## 2. v1 クラスインポートステートメントの検索ファイル

アプリケーション内のファイルをスキャンして、v1 インポートで使用される SERVICE\_IDs を探すと、使用される一意の SERVICE\_IDs が見つかります。SERVICE\_ID は、の短い一意の名前で AWS のサービス。例えば、cognitoidentityは Amazon Cognito ID の SERVICE\_ID です。

## 3. v1 インポートステートメントから v2 Maven の依存関係を確認する

すべての一意の v1 SERVICE\_IDsが見つかったら、[を参照して v2 依存関係に対応する Maven アーティファクトを決定できます](#)[the section called “パッケージ名から artifactId へのマッピング”](#)。

## 4. POM ファイルに v2 依存関係要素を追加する

ステップ 3 で決定した依存関係要素で Maven POM ファイルを更新します。

## 5. Java ファイルでは、v1 クラスから v2 クラスに段階的に変更します。

v1 クラスを v2 クラスに置き換えるときは、コンストラクタの代わりにビルダーを使用する、および fluent ゲッターとセッターを使用するなど、v2 API をサポートするために必要な変更を加えます。

## 6. POM から v1 Maven の依存関係を削除し、ファイルから v1 のインポートを削除する

v2 クラスを使用するようにコードを移行したら、残った v1 インポートをファイルから削除し、ビルドファイルからすべての依存関係を削除します。

## 7. v2 API の機能強化を使用するようにコードをリファクタリングする

コードが正常にコンパイルされ、テストに合格したら、別の HTTP クライアントやページネーターを使用してコードを簡素化するなどの v2 の機能強化を利用できます。これは任意の手順です。

## 移行例

この例では、SDK for Java v1 を使用し、複数のにアクセスするアプリケーションを移行します AWS のサービス。ステップ 5 では、次の v1 メソッドについて詳しく説明します。これは、8 つのメソッドを含むクラスの 1 つのメソッドで、アプリケーションには 32 のクラスがあります。



## v1 の移行方法

Java ファイルから v1 SDK インポートのみが以下にリストされています。

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesResult;
import com.amazonaws.services.ec2.model.Instance;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Reservation;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;
...
private static List<Instance> getRunningInstances(AmazonEC2Client ec2, List<String>
instanceIds) {
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = new DescribeInstancesRequest()
            .withInstanceIds(instanceIds);
        DescribeInstancesResult result;
        do {
            // DescribeInstancesResponse is a paginated response, so use tokens with
multiple requests.
            result = ec2.describeInstances(request);
            request.setNextToken(result.getNextToken()); // Prepare request for next
page.

            for (final Reservation r : result.getReservations()) {
                for (final Instance instance : r.getInstances()) {
                    LOGGER.info("Examining instanceId: "+ instance.getInstanceId());
                    // if instance is in a running state, add it to runningInstances
list.

                    if (RUNNING_STATES.contains(instance.getState().getName())) {
                        runningInstances.add(instance);
                    }
                }
            }
        } while (result.getNextToken() != null);
    } catch (final AmazonEC2Exception exception) {
        // if instance isn't found, assume its terminated and continue.
    }
}
```

```
        if (exception.getErrorCode().equals(NOT_FOUND_ERROR_CODE)) {
            LOGGER.info("Instance probably terminated; moving on.");
        } else {
            throw exception;
        }
    }
    return runningInstances;
}
```

## 1. v2 Maven BOM の追加

SDK for Java 2.x の Maven BOM を POM に追加し、dependencyManagementセクション内の他の依存関係も追加します。POM ファイルに SDK の v1 用の BOM がある場合は、今のところ残してください。これは後のステップで削除されます。

### POM 依存関係管理

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.example</groupId>           <!--Existing dependency in POM. -->
      <artifactId>bom</artifactId>
      <version>1.3.4</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    ...
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId> <!--Existing v1 BOM dependency. -->
      <version>1.11.1000</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    ...
    <dependency>
      <groupId>software.amazon.awssdk</groupId> <!--Add v2 BOM dependency. -->
      <artifactId>bom</artifactId>
      <version>2.24.3</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

```
</dependencyManagement>
```

## 2. v1 クラスインポートステートメントの検索ファイル

アプリケーションのコードでの一意の出現を検索します `import com.amazonaws.services`。これにより、プロジェクトで使用される v1 の依存関係を特定できます。アプリケーションに v1 の依存関係がリストされた Maven POM ファイルがある場合は、代わりにこの情報を使用できます。

この例では、[ripgrep \(rg\)](#) コマンドを使用してコードベースを検索します。

コードベースのルートから、次の `ripgrep` コマンドを実行します。がインポートステートメント `ripgrep` を検索すると、`cut`、`sort`、および `uniq` コマンドにパイプされ、SERVICE\_IDs。

```
rg --no-filename 'import\s+com\.amazonaws\.services' | cut -d '.' -f 4 | sort | uniq
```

このアプリケーションでは、次の SERVICE\_IDs がコンソールに記録されます。

```
autoscaling
cloudformation
ec2
identitymanagement
```

これは、`import` ステートメントで次の各パッケージ名が少なくとも 1 回使用されたことを示します。目的は、個々のクラス名は重要ではありません。使用する SERVICE\_IDs。

```
com.amazonaws.services.autoscaling.*
com.amazonaws.services.cloudformation.*
com.amazonaws.services.ec2.*
com.amazonaws.services.identitymanagement.*
```

## 3. v1 インポートステートメントから v2 Maven の依存関係を確認する

ステップ 2 から分離した v1 の SERVICE\_IDs。例えば `cloudformation`、`autoscaling` とは、ほとんどの場合、同じ v2 SERVICE\_ID にマッピングできます。ほとんどの場合、v2 Maven artifactId は SERVICE\_ID と一致するため、POM ファイルに依存関係ブロックを追加するために必要な情報があります。

次の表は、v2 の依存関係を確認する方法を示しています。

v1 SERVICE_ID は にマッピングされます。 パッケージ名	v2 SERVICE_ID は にマッピングされます。 パッケージ名	v2 Maven の依存関係
ec2  com.amazonaws.services. <b>ec2</b> .*	ec2  software.amazon.awssdk.services. <b>ec2</b> .*	<pre>&lt;dependency&gt;   &lt;groupId&gt;software.amazon.awssdk&lt;/groupId&gt;   &lt;artifactId&gt; <b>ec2</b>&lt;/artifactId&gt; &lt;/dependency&gt;</pre>
自動スケーリング  com.amazonaws.services. <b>autoscaling</b> .*	自動スケーリング  software.amazon.awssdk.services. <b>autoscaling</b> .*	<pre>&lt;dependency&gt;   &lt;groupId&gt;software.amazon.awssdk&lt;/groupId&gt;   &lt;artifactId&gt; <b>autoscaling</b> &lt;/artifactId&gt; &lt;/dependency&gt;</pre>
cloudformation  com.amazonaws.services. <b>cloudformation</b> .*	cloudformation  software.amazon.awssdk. <b>cloudformation</b> .*	<pre>&lt;dependency&gt;   &lt;groupId&gt;software.amazon.awssdk&lt;/groupId&gt;   &lt;artifactId&gt; <b>cloudformation</b> &lt;/artifactId&gt; &lt;/dependency&gt;</pre>
ID 管理*  com.amazonaws.services. <b>identitymanagement</b> .*	iam*  software.amazon.awssdk. <b>iam</b> .*	<pre>&lt;dependency&gt;   &lt;groupId&gt;software.amazon.awssdk&lt;/groupId&gt;   &lt;artifactId&gt; <b>iam</b>&lt;/artifactId&gt; &lt;/dependency&gt;</pre>

\* から `identitymanagement` への `iam` マッピングは、`SERVICE_ID` がバージョン間で異なる例外です。Maven または Gradle が v2 の依存関係を解決できない場合の例外 [the section called “パッケージ名から artifactId へのマッピング”](#) については、「」を参照してください。

#### 4. POM ファイルに v2 依存関係要素を追加する

ステップ 3 では、POM ファイルに追加する必要がある 4 つの依存関係ブロックを決定しました。ステップ 1 で BOM を指定しているため、バージョンを追加する必要はありません。インポートが追加されると、POM ファイルには次の依存関係要素があります。

```
...
<dependencies>
  ...
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>autoscaling</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>iam</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>cloudformation</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>ec2</artifactId>
  </dependency>
  ...
</dependencies>
...
```

#### 5. Java ファイルでは、v1 クラスから v2 クラスに段階的に変更します。

移行するメソッドには、

- からの EC2 サービスクライアント `com.amazonaws.services.ec2.AmazonEC2Client`。
- いくつかの EC2 モデルクラスが使用されています。例えば、`DescribeInstancesRequest` と `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.getInstanceIds()) {
                    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;
}
...
```

すべての v1 インポートを v2 インポートに置き換えることが目標です。一度に 1 つのクラスに進みます。

#### a. インポートステートメントまたはクラス名を置き換える

`describeRunningInstances` メソッドの最初のパラメータは v1 `AmazonEC2Client` インスタンスであることがわかります。次のいずれかを行います。

- のインポートを `com.amazonaws.services.ec2.AmazonEC2Client` に置き換え `software.amazon.awssdk.services.ec2.Ec2Client`、`AmazonEC2Client` を に変更します `Ec2Client`。
- パラメータタイプを に変更 `Ec2Client` し、IDE に正しいインポートを求めるようにします。IDE では、クライアント名が異なるため、v2 クラスをインポートするように求められます。`AmazonEC2Client` と です `Ec2Client`。クラス名が両方のバージョンで同じ場合、このアプローチは機能しません。

#### b. v1 モデルクラスを同等の v2 に置き換える

v2 への変更後 `Ec2Client`、IDE を使用すると、次のステートメントにコンパイルエラーが表示されます。

```
result = ec2.describeInstances(request);
```

コンパイルエラーは、v1 の のインスタンスを v2 `Ec2Client` `describeInstances` メソッドのパラメータ `DescribeInstancesRequest` として使用した結果です。修正するには、次の置換ステートメントまたはインポートステートメントを実行します。

replace	with
<pre>import com.amazonaws.services.ec2.model.DescribeInstancesRequest</pre>	<pre>import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest</pre>

c. v1 コンストラクタを v2 ビルダーに変更します。

v2 クラスにはコンストラクタがないため、コンパイルエラーが表示されます。修正するには、次の変更を行います。

変更	から
<pre>final DescribeInstancesRequest request = new DescribeInstancesRequest().withInstanceIds(instanceIdsCopy);</pre>	<pre>final DescribeInstancesRequest request = DescribeInstancesRequest.builder().instanceIds(instanceIdsCopy).build();</pre>

d. v1 **\*Result**レスポンスオブジェクトを**\*Response**同等の v2 に置き換える

v1 と v2 の一貫した違いは、v2 のすべてのレスポンスオブジェクトが \*Responseではなくで終わる\*Resultことです。v2 DescribeInstancesResultインポート への v1 インポートを置き換えますDescribeInstancesResponse。

d. API の変更

次のステートメントにはいくつかの変更が必要です。

```
request.setNextToken(result.getNextToken());
```

v2 では、セッターメソッドは `set`または `setPrefix` を使用しません。プレフィックスが付いた Getter メソッド `get`は SDK for Java 2.x にも記載されています。

`request` インスタンスなどのモデルクラスは v2 ではイミュータブルであるため、ビルダー `DescribeInstancesRequest`を使用して新しいを作成する必要があります。



v2 では、ステートメントは次のようになります。

```
request = DescribeInstancesRequest.builder()
    .nextToken(result.nextToken())
    .build();
```

d. v2 クラスでメソッドがコンパイルされるまで繰り返します。

コードの残りの部分に進みます。v1 imports を v2 imports に置き換え、コンパイルエラーを修正します。必要に応じて、[v2 API リファレンス](#)と「[別のリファレンス](#)」を参照してください。

この単一のメソッドを移行した後、次の v2 コードがあります。

```
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;
}
...

```

8つのメソッドを持つJavaファイル内の1つのメソッドを移行するため、ファイルを操作するときにv1とv2のインポートが混在します。ステップの実行時に、最後の6つのインポートステートメントを追加しました。

すべてのコードを移行すると、v1インポートステートメントはこれ以上なくなります。

## 6. POM から v1 Maven の依存関係を削除し、ファイルから v1 のインポートを削除する

ファイル内のすべてのv1コードを移行した後、次のv2 SDKインポートステートメントがありません。

```
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;
```

アプリケーション内のすべてのファイルを移行すると、POM ファイル内の v1 依存関係は不要になります。を使用している場合は、`dependencyManagement` セクションから v1 BOM を削除し、すべての v1 依存関係ブロックを削除します。

## 7. v2 API の機能強化を使用するようにコードをリファクタリングする

移行中のスニペットについては、オプションで v2 ページネーターを使用し、より多くのデータに対するトークンベースのリクエストを SDK に管理させることができます。

do 句全体を次のように置き換えることができます。

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

## パッケージ名から Maven artifactId へのマッピング

Maven または Gradle プロジェクトを SDK for Java の v1 から v2 に移行するときは、ビルドファイルに追加する依存関係を特定する必要があります。[the section called “Step-by-step 命令”](#) (ステップ 3) で説明されているアプローチでは、インポートステートメントのパッケージ名を出発点として使用し、ビルドファイルに追加する依存関係 (アーティファクト ID を決定します)。

このトピックの情報をを使用して、v1 パッケージ名を v2 artifactIds にマッピングできます。

## パッケージ名と Maven artifactIds で使用される一般的な命名規則

次の表は、SDKs が特定の SERVICE\_ID に使用する一般的な命名規則を示しています。SERVICE\_ID は、の一意の識別子です AWS のサービス。例えば、Amazon S3 サービスの SERVICE\_ID は s3 であり、cognitoidentity は Amazon Cognito ID の SERVICE\_ID です。

v1 パッケージ名 (インポートステートメント)	v1 artifactId	v2 artifactId	v2 パッケージ名 (インポートステートメント)
com.amazonaws.services.SERVICE_ID	aws-java-sdk-SERVICE_ID	SERVICE_ID	software.amazon.awssdk.services.SERVICE_ID

Amazon Cognito ID の例 (SERVICE\_ID: **cognitoidentity** )

com.amazonaws.services.cognitoidentity	aws-java-sdk- 認識	認識	software.amazon.awssdk.services.cognitoidentity
--	------------------	----	---

## SERVICE\_ID の違い

### v1 内

場合によっては、service\_ID がパッケージ名と同じサービスの artifactId で異なることがあります。例えば、次の表の CloudWatch メトリクス行は、パッケージ名には metrics が SERVICE\_ID ですが、アーartifacldcloudwatchmetrics は の SERVICE\_ID であることを示しています。

### v2 内

パッケージ名と artifactIds に違いはありません。

### v1 と v2 の間

ほとんどのサービスでは、v2 の SERVICE\_ID はパッケージ名と artifactIds。この例が、前の表で示した cognitoidentity SERVICE\_ID です。ただし、一部の SERVICE\_IDs は、次の表に示すように SDKs 間で異なります。

いずれかの v1 列の太字の SERVICE\_ID は、v2 で使用される SERVICE\_ID とは異なることを示します。

サービス名	v1 パッケージ名	v1 artifactId	v2 artifactId	v2 パッケージ名
	すべてのパッケージ名は <code>com.amazonaws.services</code> 、最初の行に示すように始まります。	すべての artifactIds は、最初の行に示すようにタグで囲まれます。	すべての artifactIds は、最初の行に示すようにタグで囲まれます。	すべてのパッケージ名は <code>software.amazon.awssdk</code> 、最初の行に示すように始まります。
API Gateway	<code>com.amazonaws.services.apigateway</code>	<code>&lt;artifactId&gt;aws-java-sdk-apigateway &lt;/artifactId&gt;</code>	<code>&lt;artifactId&gt;apigateway&lt;/artifactId&gt;</code>	<code>software.amazon.awssdk.services.apigateway</code>
アプリレジストリ	アプリレジストリ	アプリレジストリ	<code>servicecatalogappregistry</code>	<code>servicecatalogappregistry</code>
Application Discovery	アプリケーション検出	discovery	アプリケーション検出	アプリケーション検出
拡張 AI ランタイム	拡張された <code>airuntime</code>	拡張された <code>airuntime</code>	<code>sagemaker a2iruntime</code>	<code>sagemaker a2iruntime</code>
Certificate Manager	CertificateManager	acm	acm	acm
CloudControl API	クラウドコントロール	クラウドコントロール	クラウドコントロール	クラウドコントロール
CloudSearch	<code>cloudsearchv2</code>	<code>cloudsearch</code>	<code>cloudsearch</code>	<code>cloudsearch</code>
CloudSearch ドメイン	<code>cloudsearchdomain</code>	クラウドサーチ	<code>cloudsearchdomain</code>	<code>cloudsearchdomain</code>

サービス名	v1 パッケージ名	v1 artifactId	v2 artifactId	v2 パッケージ名
CloudWatch イベント	cloudwatchevents	イベント	cloudwatchevents	cloudwatchevents
CloudWatch Evidently	cloudwatchevidently	cloudwatchevidently	evidently	evidently
CloudWatch ログ	ログ	ログ	cloudwatchlogs	cloudwatchlogs
CloudWatch メトリクス	メトリクス	cloudwatchmetrics	cloudwatch	cloudwatch
CloudWatch Rum	cloudwatchrum	cloudwatchrum	rum	rum
Cognito ID プロバイダー	cognitoidp	cognitoidp	匿名プロバイダー	匿名プロバイダー
Connect キャンペーン	接続キャンペーン	接続キャンペーン	接続キャンペーン	接続キャンペーン
Wisdom を接続する	connectwisdom	connectwisdom	wisdom	wisdom
Database Migration Service	データベース移行サービス	dms	データベース移行	データベース移行
DataZone	データゾーン	データゾーン外部	データゾーン	データゾーン
DynamoDB	dynamodbv2	dynamodb	dynamodb	dynamodb
Elastic ファイルシステム	エラスティックファイルシステム	efs	efs	efs

サービス名	v1 パッケージ名	v1 artifactId	v2 artifactId	v2 パッケージ名
Elastic Map Reduce	elasticmapreduce	EMR	EMR	EMR
Glue DataBrew	グルダタブリ	グルダタブリ	databrew	databrew
IAM Roles Anywhere	iamroles 任意の場所	iamroles 任意の場所	rolesanywhere	rolesanywhere
ID 管理	ID 管理	iam	iam	iam
IoT データ	iotdata	iot	iotdataplane	iotdataplane
Kinesis Analytics	kinesisanalytics	kinesis	kinesisanalytics	kinesisanalytics
Kinesis Firehose	Kinesisfirehose	kinesis	firehose	firehose
Kinesis Video Signaling チャネル	Kinesis Videosignaling チャネル	Kinesis Videosignaling チャネル	Kinesis Videosignaling	Kinesis Videosignaling
Lex	lexruntime	lex	lexruntime	lexruntime
Lookout for Vision	ルックアウトプロビジョニング	ルックアウトプロビジョニング	lookoutvision	lookoutvision
Mainframe Modernization	メインフレームモダナイゼーション	メインフレームモダナイゼーション	m2	m2
Marketplace Metering	マーケットプレイス計測	マーケットプレイス計測サービス	マーケットプレイス計測	マーケットプレイス計測
マネージド Grafana	Managedgrafana	Managedgrafana	grafana	grafana
Mechanical Turk	mturk	テクスチャタークリックエスタ	mturk	mturk

サービス名	v1 パッケージ名	v1 artifactId	v2 artifactId	v2 パッケージ名
Migration Hub Strategy の推奨事項	migration hubstrate gyrecomme ndations	migration hubstrate gyrecomme ndations	移行ハブレー ジー	移行ハブレー ジー
Nimble Studio	nimblestudio	nimblestudio	nimble	nimble
プライベート 5G	private5g	private5g	プライベート ネットワーク	プライベート ネットワーク
Prometheus	Prometheus	Prometheus	amp	amp
ごみ箱	ごみ箱	ごみ箱	rbin	rbin
Redshift Data API	redshiftdataapi	redshiftdataapi	redshiftdata	redshiftdata
Route 53	route53 ドメイン	route53	route53 ドメイン	route53 ドメイン
Sage Maker Edge Manager	sagemaker edgemanager	sagemaker edgemanager	sagemakeredge	sagemakeredge
セキュリティ トークン	セキュリティ トークン	sts	sts	sts
サーバーの移行	サーバー移行	サーバー移行	sms	sms
シンプルな E メール	simpleemail	ses	ses	ses
シンプルな E メール V2	simpleemailv2	sesv2	sesv2	sesv2
シンプルなシス テム管理	シンプルなシス テム管理	ssm	ssm	ssm
シンプルなワー クフロー	シンプルなワー クフロー	シンプルなワー クフロー	swf	swf



サービス名	v1 パッケージ名	v1 artifactId	v2 artifactId	v2 パッケージ名
Step Functions	stepfunctions	stepfunctions	sfn	sfn

## AWS SDK for Java 1.x と 2.x の違い

このセクションでは、アプリケーションをバージョン 1.x から AWS SDK for Java バージョン 2.x に変換する際に注意すべき主な変更について説明します。

### パッケージ名の変更

SDK for Java 1.x から SDK for Java 2.x への主な変更点は、パッケージ名の変更です。SDK 2.x ではパッケージ名が `software.amazon.awssdk` で始まりますが、SDK 1.x では `com.amazonaws` で始まります。

これらの同じ名前は、SDK 1.x から SDK 2.x までの Maven アーティファクトを区別します。SDK 2.x の Maven アーティファクトは `software.amazon.awssdk groupId` を使用しますが、SDK 1.x では `com.amazonaws groupId` を使用します。

SDK 2.x のアーティファクトのみを使用するプロジェクトの `com.amazonaws` 依存関係がコードに必要な場合があります。その一例が、サーバーサイド AWS Lambda で使用する場合があります。これは、このガイドの前半の「[Apache Maven プロジェクトのセットアップ](#)」セクションで説明しました。

#### Note

SDK 1.x のいくつかのパッケージ名には、v2 を含みます。この場合の v2 の使用は、通常、パッケージ内のコードがバージョン 2 のサービスで動作するように設定されていることを意味します。

フルパッケージ名は `com.amazonaws` で始まるため、これらは SDK 1.x コンポーネントです。SDK 1.x のパッケージ名の例は次のとおりです。

- `com.amazonaws.services.dynamodbv2`
- `com.amazonaws.retry.v2`
- `com.amazonaws.services.apigatewayv2`
- `com.amazonaws.services.simpleemailv2`

## プロジェクトへのバージョン 2.x の追加

Maven は、AWS SDK for Java 2.x を使用するときには依存関係を管理するための推奨される方法です。バージョン 2.x コンポーネントをプロジェクトに追加するには、SDK への依存関係で `pom.xml` ファイルを更新します。

### Example

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.16.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>dynamodb</artifactId>
  </dependency>
</dependencies>
```

プロジェクトを [バージョン 2.x に移行するとき](#)、[バージョン 1.x と side-by-side 2.x](#) を使用することもできます。

## イミュータブルな POJO

クライアントとオペレーションリクエスト、および応答オブジェクトがイミュータブルになり、作成後に変更できなくなりました。リクエストあるいはレスポンス変数を再利用するには、新しいオブジェクトを構築してそれを割り当てる必要があります。

### Example 1.x でリクエストオブジェクトを更新する場合

```
DescribeAlarmsRequest request = new DescribeAlarmsRequest();
DescribeAlarmsResult response = cw.describeAlarms(request);

request.setNextToken(response.getNextToken());
```

## Example 2.x でリクエストオブジェクトを更新する場合

```
DescribeAlarmsRequest request = DescribeAlarmsRequest.builder().build();
DescribeAlarmsResponse response = cw.describeAlarms(request);

request = DescribeAlarmsRequest.builder()
    .nextToken(response.nextToken())
    .build();
```

## セッターメソッドとゲッターメソッド

AWS SDK for Java 2.x では、セッターメソッド名に `set` または `with` プレフィックスは含まれません。たとえば、`*.withEndpoint()` は現在 `*.endpoint()` です。

Getter メソッド名は `get` プレフィックスを使用しません。

### Example 1.x でセッターメソッドを使用する

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
    .withRegion("us-east-1")
    .build();
```

### Example 2.x でセッターメソッドを使用する

```
DynamoDbClient client = DynamoDbClient.builder()
    .region(Region.US_EAST_1)
    .build();
```

### Example 1.x でゲッターメソッドを使用する

```
String token = request.getNextToken();
```

### Example 2.x でゲッターメソッドを使用する

```
String token = request.nextToken();
```

## モデルクラス名

サービスレスポンスを表すモデルクラス名は、v1 が使用する `Result` のではなく、v2 `Response` で終わります。

## Example v1 のレスポンスを表すクラス名の

```
CreateApiKeyResult
AllocateAddressResult
```

## Example v2 のレスポンスを表すクラス名の

```
CreateApiKeyResponse
AllocateAddressResponse
```

## ライブラリとユーティリティの移行状況

## SDK for Java ライブラリとユーティリティ

次の表に、SDK for Java のライブラリとユーティリティの移行ステータスを示します。

バージョン 1.12.x の名前	バージョン 2.x の名前	バージョン 2.x 以降
DynamoDBMapper	<a href="#">DynamoDbEnhancedClient</a>	2.12.0
ウェイター	<a href="#">ウェーター</a>	2.15.0
CloudFrontUrlSigner, CloudFrontCookieSigner	<a href="#">CloudFrontUtilities</a>	2.18.33
TransferManager	<a href="#">S3TransferManager</a>	2.19.0
EC2 メタデータクライアント	<a href="#">EC2 メタデータクライアント</a>	2.19.29
S3 URI パーサー	<a href="#">S3 URI パーサー</a>	2.20.41
IAM Policy Builder	<a href="#">IAM Policy Builder</a>	2.20.126
Amazon SQS のクライアント 側のバッファリング	自動リクエストバッチ処理	<a href="#">未リリース</a>
進行状況のリスナー	進行状況のリスナー	<a href="#">未リリース</a>

## 関連ライブラリ

次の表は、個別にリリースされているものの、SDK for Java 2.x で動作するライブラリの一覧です。

SDK for Java のバージョン 2.x で使用されている名前	バージョン以降
<a href="#">Amazon S3 暗号化クライアント</a>	3.0.0 <sup>1</sup>
<a href="#">AWS DynamoDB 用のデータベース暗号化クライアント</a>	3.0.0 <sup>2</sup>

<sup>1</sup> Amazon S3 の暗号化クライアントは、次の Maven 依存関係を使用して利用できます。

```
<dependency>
  <groupId>software.amazon.encryption.s3</groupId>
  <artifactId>amazon-s3-encryption-client-java</artifactId>
  <version>3.x</version>
</dependency>
```

<sup>2</sup> DynamoDB 用の AWS Database Encryption Client は、次の Maven 依存関係を使用して使用できません。

```
<dependency>
  <groupId>software.amazon.cryptography</groupId>
  <artifactId>aws-database-encryption-sdk-dynamodb</artifactId>
  <version>3.x</version>
</dependency>
```

## ライブラリとユーティリティの移行の詳細

- [S3 Transfer Manager](#)
- [EC2 メタデータユーティリティ](#)
- [CloudFront 事前署名](#)
- [S3 URI 解析](#)

## クライアントの変更

### クライアントビルダー

すべてのクライアントは、クライアントビルダーメソッドを使用して作成する必要があります。コンストラクタは利用できなくなりました。

Example バージョン 1.x でクライアントを作成する場合

```
AmazonDynamoDB ddbClient = AmazonDynamoDBClientBuilder.defaultClient();
AmazonDynamoDBClient ddbClient = new AmazonDynamoDBClient();
```

Example バージョン 2.x でクライアントを作成する場合

```
DynamoDbClient ddbClient = DynamoDbClient.create();
DynamoDbClient ddbClient = DynamoDbClient.builder().build();
```

### クライアントクラス名

すべてのクライアント名は完全キャメルケースになり、Amazon によるプレフィックスは使用されなくなりました。上記の変更は、AWS CLIで使用される名前と連携します。

Example 1.x のクラス名の場合

```
AmazonDynamoDB
AWSACMPCAAsyncClient
```

Example 2.x のクラス名の場合

```
DynamoDbClient
AcmAsyncClient
```

### クライアントクラス名の変更

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.acmpca.AWSACMPCAAsyncClient</code>	<code>software.amazon.awssdk.services.acm.AcmAsyncClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.acmpca.AWSACMPCAClient</code>	<code>software.amazon.awssdk.services.acm.AcmClient</code>
<code>com.amazonaws.services.alexforbusiness.AmazonAlexaForBusinessAsyncClient</code>	<code>software.amazon.awssdk.services.alexforbusiness.AlexaForBusinessAsyncClient</code>
<code>com.amazonaws.services.alexforbusiness.AmazonAlexaForBusinessClient</code>	<code>software.amazon.awssdk.services.alexforbusiness.AlexaForBusinessClient</code>
<code>com.amazonaws.services.apigateway.AmazonApiGatewayAsyncClient</code>	<code>software.amazon.awssdk.services.apigateway.ApiGatewayAsyncClient</code>
<code>com.amazonaws.services.apigateway.AmazonApiGatewayClient</code>	<code>software.amazon.awssdk.services.apigateway.ApiGatewayClient</code>
<code>com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingAsyncClient</code>	<code>software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingAsyncClient</code>
<code>com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingClient</code>	<code>software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient</code>
<code>com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryAsyncClient</code>	<code>software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryAsyncClient</code>
<code>com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryClient</code>	<code>software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.appstream.AmazonAppStreamAsyncClient</code>	<code>software.amazon.awssdk.services.appstream.AppStreamAsyncClient</code>
<code>com.amazonaws.services.appstream.AmazonAppStreamClient</code>	<code>software.amazon.awssdk.services.appstream.AppStreamClient</code>
<code>com.amazonaws.services.appsync.AWSAppSyncAsyncClient</code>	<code>software.amazon.awssdk.services.appsync.AppSyncAsyncClient</code>
<code>com.amazonaws.services.appsync.AWSAppSyncClient</code>	<code>software.amazon.awssdk.services.appsync.AppSyncClient</code>
<code>com.amazonaws.services.athena.AmazonAthenaAsyncClient</code>	<code>software.amazon.awssdk.services.athena.AthenaAsyncClient</code>
<code>com.amazonaws.services.athena.AmazonAthenaClient</code>	<code>software.amazon.awssdk.services.athena.AthenaClient</code>
<code>com.amazonaws.services.autoscaling.AmazonAutoScalingAsyncClient</code>	<code>software.amazon.awssdk.services.autoscaling.AutoScalingAsyncClient</code>
<code>com.amazonaws.services.autoscaling.AmazonAutoScalingClient</code>	<code>software.amazon.awssdk.services.autoscaling.AutoScalingClient</code>
<code>com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansAsyncClient</code>	<code>software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansAsyncClient</code>
<code>com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansClient</code>	<code>software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansClient</code>
<code>com.amazonaws.services.batch.AWSBatchAsyncClient</code>	<code>software.amazon.awssdk.services.batch.BatchAsyncClient</code>



1.x クライアント	2.x クライアント
<code>com.amazonaws.services.batch.AWSBatchClient</code>	<code>software.amazon.awssdk.services.batch.BatchClient</code>
<code>com.amazonaws.services.budgets.AWSBudgetsAsyncClient</code>	<code>software.amazon.awssdk.services.budgets.BudgetsAsyncClient</code>
<code>com.amazonaws.services.budgets.AWSBudgetsClient</code>	<code>software.amazon.awssdk.services.budgets.BudgetsClient</code>
<code>com.amazonaws.services.certificatemanager.AWSCertificateManagerAsyncClient</code>	<code>software.amazon.awssdk.services.acm.AcmAsyncClient</code>
<code>com.amazonaws.services.certificatemanager.AWSCertificateManagerClient</code>	<code>software.amazon.awssdk.services.acm.AcmClient</code>
<code>com.amazonaws.services.cloud9.AWSCloud9AsyncClient</code>	<code>software.amazon.awssdk.services.cloud9.Cloud9AsyncClient</code>
<code>com.amazonaws.services.cloud9.AWSCloud9Client</code>	<code>software.amazon.awssdk.services.cloud9.Cloud9Client</code>
<code>com.amazonaws.services.clouddirectory.AmazonCloudDirectoryAsyncClient</code>	<code>software.amazon.awssdk.services.clouddirectory.CloudDirectoryAsyncClient</code>
<code>com.amazonaws.services.clouddirectory.AmazonCloudDirectoryClient</code>	<code>software.amazon.awssdk.services.clouddirectory.CloudDirectoryClient</code>
<code>com.amazonaws.services.cloudformation.AmazonCloudFormationAsyncClient</code>	<code>software.amazon.awssdk.services.cloudformation.CloudFormationAsyncClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.cloudformation.AmazonCloudFormationClient</code>	<code>software.amazon.awssdk.services.cloudformation.CloudFormationClient</code>
<code>com.amazonaws.services.cloudfront.AmazonCloudFrontAsyncClient</code>	<code>software.amazon.awssdk.services.cloudfront.CloudFrontAsyncClient</code>
<code>com.amazonaws.services.cloudfront.AmazonCloudFrontClient</code>	<code>software.amazon.awssdk.services.cloudfront.CloudFrontClient</code>
<code>com.amazonaws.services.cloudhsm.AWSCloudHSMAsyncClient</code>	<code>software.amazon.awssdk.services.cloudhsm.CloudHsmAsyncClient</code>
<code>com.amazonaws.services.cloudhsm.AWSCloudHSMClient</code>	<code>software.amazon.awssdk.services.cloudhsm.CloudHsmClient</code>
<code>com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2AsyncClient</code>	<code>software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2AsyncClient</code>
<code>com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2Client</code>	<code>software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2Client</code>
<code>com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainAsyncClient</code>	<code>software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainAsyncClient</code>
<code>com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainClient</code>	<code>software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.cloudsearchv2.AmazonCloudSearchAsyncClient</code>	<code>software.amazon.awssdk.services.cloudsearch.CloudSearchAsyncClient</code>
<code>com.amazonaws.services.cloudsearchv2.AmazonCloudSearchClient</code>	<code>software.amazon.awssdk.services.cloudsearch.CloudSearchClient</code>
<code>com.amazonaws.services.cloudtrail.AWSCloudTrailAsyncClient</code>	<code>software.amazon.awssdk.services.cloudtrail.CloudTrailAsyncClient</code>
<code>com.amazonaws.services.cloudtrail.AWSCloudTrailClient</code>	<code>software.amazon.awssdk.services.cloudtrail.CloudTrailClient</code>
<code>com.amazonaws.services.cloudwatch.AmazonCloudWatchAsyncClient</code>	<code>software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient</code>
<code>com.amazonaws.services.cloudwatch.AmazonCloudWatchClient</code>	<code>software.amazon.awssdk.services.cloudwatch.CloudWatchClient</code>
<code>com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsAsyncClient</code>	<code>software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsAsyncClient</code>
<code>com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClient</code>	<code>software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient</code>
<code>com.amazonaws.services.codebuild.AWSCodeBuildAsyncClient</code>	<code>software.amazon.awssdk.services.codebuild.CodeBuildAsyncClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.codebuild.AWSCodeBuildClient</code>	<code>software.amazon.awssdk.services.codebuild.CodeBuildClient</code>
<code>com.amazonaws.services.codecommit.AWSCodeCommitAsyncClient</code>	<code>software.amazon.awssdk.services.codecommit.CodeCommitAsyncClient</code>
<code>com.amazonaws.services.codecommit.AWSCodeCommitClient</code>	<code>software.amazon.awssdk.services.codecommit.CodeCommitClient</code>
<code>com.amazonaws.services.codedeploy.AmazonCodeDeployAsyncClient</code>	<code>software.amazon.awssdk.services.codedeploy.CodeDeployAsyncClient</code>
<code>com.amazonaws.services.codedeploy.AmazonCodeDeployClient</code>	<code>software.amazon.awssdk.services.codedeploy.CodeDeployClient</code>
<code>com.amazonaws.services.codepipeline.AWSCodePipelineAsyncClient</code>	<code>software.amazon.awssdk.services.codepipeline.CodePipelineAsyncClient</code>
<code>com.amazonaws.services.codepipeline.AWSCodePipelineClient</code>	<code>software.amazon.awssdk.services.codepipeline.CodePipelineClient</code>
<code>com.amazonaws.services.codestar.AWSCodeStarAsyncClient</code>	<code>software.amazon.awssdk.services.codestar.CodeStarAsyncClient</code>
<code>com.amazonaws.services.codestar.AWSCodeStarClient</code>	<code>software.amazon.awssdk.services.codestar.CodeStarClient</code>
<code>com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityAsyncClient</code>	<code>software.amazon.awssdk.services.cognitoidentity.CognitoIdentityAsyncClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityClient</code>	<code>software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient</code>
<code>com.amazonaws.services.cognitoidp.AWSCognitoIdentityProviderAsyncClient</code>	<code>software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderAsyncClient</code>
<code>com.amazonaws.services.cognitoidp.AWSCognitoIdentityProviderClient</code>	<code>software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient</code>
<code>com.amazonaws.services.cognitosync.AmazonCognitoSyncAsyncClient</code>	<code>software.amazon.awssdk.services.cognitosync.CognitoSyncAsyncClient</code>
<code>com.amazonaws.services.cognitosync.AmazonCognitoSyncClient</code>	<code>software.amazon.awssdk.services.cognitosync.CognitoSyncClient</code>
<code>com.amazonaws.services.comprehend.AmazonComprehendAsyncClient</code>	<code>software.amazon.awssdk.services.comprehend.ComprehendAsyncClient</code>
<code>com.amazonaws.services.comprehend.AmazonComprehendClient</code>	<code>software.amazon.awssdk.services.comprehend.ComprehendClient</code>
<code>com.amazonaws.services.config.AmazonConfigAsyncClient</code>	<code>software.amazon.awssdk.services.config.ConfigAsyncClient</code>
<code>com.amazonaws.services.config.AmazonConfigClient</code>	<code>software.amazon.awssdk.services.config.ConfigClient</code>
<code>com.amazonaws.services.connect.AmazonConnectAsyncClient</code>	<code>software.amazon.awssdk.services.connect.ConnectAsyncClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.connect.AmazonConnectClient</code>	<code>software.amazon.awssdk.services.connect.ConnectClient</code>
<code>com.amazonaws.services.costandusagereport.AWSCostAndUsageReportAsyncClient</code>	<code>software.amazon.awssdk.services.costandusagereport.CostAndUsageReportAsyncClient</code>
<code>com.amazonaws.services.costandusagereport.AWSCostAndUsageReportClient</code>	<code>software.amazon.awssdk.services.costandusagereport.CostAndUsageReportClient</code>
<code>com.amazonaws.services.costexplorer.AWSCostExplorerAsyncClient</code>	<code>software.amazon.awssdk.services.costexplorer.CostExplorerAsyncClient</code>
<code>com.amazonaws.services.costexplorer.AWSCostExplorerClient</code>	<code>software.amazon.awssdk.services.costexplorer.CostExplorerClient</code>
<code>com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceAsyncClient</code>	<code>software.amazon.awssdk.services.databasemigration.DatabaseMigrationAsyncClient</code>
<code>com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceClient</code>	<code>software.amazon.awssdk.services.databasemigration.DatabaseMigrationClient</code>
<code>com.amazonaws.services.datapipeline.DataPipelineAsyncClient</code>	<code>software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient</code>
<code>com.amazonaws.services.datapipeline.DataPipelineClient</code>	<code>software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.dax. AmazonDaxAsyncClient</code>	<code>software.amazon.awssdk.serv ices.dax.DaxAsyncClient</code>
<code>com.amazonaws.services.dax. AmazonDaxClient</code>	<code>software.amazon.awssdk.serv ices.dax.DaxClient</code>
<code>com.amazonaws.services.devi cefarm.AWSDeviceFarmAsyncClient</code>	<code>software.amazon.awssdk.serv ices.devicefarm.DeviceFarmA syncClient</code>
<code>com.amazonaws.services.devi cefarm.AWSDeviceFarmClient</code>	<code>software.amazon.awssdk.serv ices.devicefarm.DeviceFarmC lient</code>
<code>com.amazonaws.services.dire ctconnect.AmazonDirectConne ctAsyncClient</code>	<code>software.amazon.awssdk.serv ices.directconnect.DirectCo nnectAsyncClient</code>
<code>com.amazonaws.services.dire ctconnect.AmazonDirectConne ctClient</code>	<code>software.amazon.awssdk.serv ices.directconnect.DirectCo nnectClient</code>
<code>com.amazonaws.services.dire ctory.AWSDirectoryServiceAs yncClient</code>	<code>software.amazon.awssdk.serv ices.directory.DirectoryAsy ncClient</code>
<code>com.amazonaws.services.dire ctory.AWSDirectoryServiceClient</code>	<code>software.amazon.awssdk.serv ices.directory.DirectoryClient</code>
<code>com.amazonaws.services.dlm. AmazonDLMAsyncClient</code>	<code>software.amazon.awssdk.serv ices.dlm.DlmAsyncClient</code>
<code>com.amazonaws.services.dlm. AmazonDLMClient</code>	<code>software.amazon.awssdk.serv ices.dlm.DlmClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBAsyncClient</code>	<code>software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBClient</code>	<code>software.amazon.awssdk.services.dynamodb.DynamoDbClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsAsyncClient</code>	<code>software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsAsyncClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsClient</code>	<code>software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsClient</code>
<code>com.amazonaws.services.ec2.AmazonEC2AsyncClient</code>	<code>software.amazon.awssdk.services.ec2.Ec2AsyncClient</code>
<code>com.amazonaws.services.ec2.AmazonEC2Client</code>	<code>software.amazon.awssdk.services.ec2.Ec2Client</code>
<code>com.amazonaws.services.ecr.AmazonECRAAsyncClient</code>	<code>software.amazon.awssdk.services.ecr.EcrAsyncClient</code>
<code>com.amazonaws.services.ecr.AmazonECRClient</code>	<code>software.amazon.awssdk.services.ecr.EcrClient</code>
<code>com.amazonaws.services.ecs.AmazonECSAsyncClient</code>	<code>software.amazon.awssdk.services.ecs.EcsAsyncClient</code>
<code>com.amazonaws.services.ecs.AmazonECSClient</code>	<code>software.amazon.awssdk.services.ecs.EcsClient</code>
<code>com.amazonaws.services.eks.AmazonEKSAAsyncClient</code>	<code>software.amazon.awssdk.services.eks.EksAsyncClient</code>



1.x クライアント	2.x クライアント
<code>com.amazonaws.services.eks. AmazonEKSClient</code>	<code>software.amazon.awssdk.serv ices.eks.EksClient</code>
<code>com.amazonaws.services.elas tocache.AmazonElasticCacheAs yncClient</code>	<code>software.amazon.awssdk.serv ices.elasticache.ElasticCach eAsyncClient</code>
<code>com.amazonaws.services.elas tocache.AmazonElasticCacheClient</code>	<code>software.amazon.awssdk.serv ices.elasticache.ElasticCach eClient</code>
<code>com.amazonaws.services.elas ticbeanstalk.AWSElasticBean stalkAsyncClient</code>	<code>software.amazon.awssdk.serv ices.elasticbeanstalk.Elast icBeanstalkAsyncClient</code>
<code>com.amazonaws.services.elas ticbeanstalk.AWSElasticBean stalkClient</code>	<code>software.amazon.awssdk.serv ices.elasticbeanstalk.Elast icBeanstalkClient</code>
<code>com.amazonaws.services.elas ticfilesystem.AmazonElastic FileSystemAsyncClient</code>	<code>software.amazon.awssdk.serv ices.efs.EfsAsyncClient</code>
<code>com.amazonaws.services.elas ticfilesystem.AmazonElastic FileSystemClient</code>	<code>software.amazon.awssdk.serv ices.efs.EfsClient</code>
<code>com.amazonaws.services.elas ticloadbalancing.AmazonElas ticLoadBalancingAsyncClient</code>	<code>software.amazon.awssdk.serv ices.elasticloadbalancing.E lasticLoadBalancingAsyncClient</code>
<code>com.amazonaws.services.elas ticloadbalancing.AmazonElas ticLoadBalancingClient</code>	<code>software.amazon.awssdk.serv ices.elasticloadbalancing.E lasticLoadBalancingClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingAsyncClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2AsyncClient</code>
<code>com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2Client</code>
<code>com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceAsyncClient</code>	<code>software.amazon.awssdk.services.emr.EmrAsyncClient</code>
<code>com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClient</code>	<code>software.amazon.awssdk.services.emr.EmrClient</code>
<code>com.amazonaws.services.elasticsearch.AWSElasticsearchAsyncClient</code>	<code>software.amazon.awssdk.services.elasticsearch.ElasticsearchAsyncClient</code>
<code>com.amazonaws.services.elasticsearch.AWSElasticsearchClient</code>	<code>software.amazon.awssdk.services.elasticsearch.ElasticsearchClient</code>
<code>com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderAsyncClient</code>	<code>software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderAsyncClient</code>
<code>com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderClient</code>	<code>software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderClient</code>
<code>com.amazonaws.services.fms.AWSFMSAsyncClient</code>	<code>software.amazon.awssdk.services.fms.FmsAsyncClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.fms.AWSFMSClient</code>	<code>software.amazon.awssdk.services.fms.FmsClient</code>
<code>com.amazonaws.services.gamelift.AmazonGameLiftAsyncClient</code>	<code>software.amazon.awssdk.services.gamelift.GameLiftAsyncClient</code>
<code>com.amazonaws.services.gamelift.AmazonGameLiftClient</code>	<code>software.amazon.awssdk.services.gamelift.GameLiftClient</code>
<code>com.amazonaws.services.glacier.AmazonGlacierAsyncClient</code>	<code>software.amazon.awssdk.services.glacier.GlacierAsyncClient</code>
<code>com.amazonaws.services.glacier.AmazonGlacierClient</code>	<code>software.amazon.awssdk.services.glacier.GlacierClient</code>
<code>com.amazonaws.services.glue.AWSGlueAsyncClient</code>	<code>software.amazon.awssdk.services.glue.GlueAsyncClient</code>
<code>com.amazonaws.services.glue.AWSGlueClient</code>	<code>software.amazon.awssdk.services.glue.GlueClient</code>
<code>com.amazonaws.services.greengrass.AWSGreengrassAsyncClient</code>	<code>software.amazon.awssdk.services.greengrass.GreengrassAsyncClient</code>
<code>com.amazonaws.services.greengrass.AWSGreengrassClient</code>	<code>software.amazon.awssdk.services.greengrass.GreengrassClient</code>
<code>com.amazonaws.services.guardduty.AmazonGuardDutyAsyncClient</code>	<code>software.amazon.awssdk.services.guardduty.GuardDutyAsyncClient</code>
<code>com.amazonaws.services.guardduty.AmazonGuardDutyClient</code>	<code>software.amazon.awssdk.services.guardduty.GuardDutyClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.health.AWSHealthAsyncClient</code>	<code>software.amazon.awssdk.services.health.HealthAsyncClient</code>
<code>com.amazonaws.services.health.AWSHealthClient</code>	<code>software.amazon.awssdk.services.health.HealthClient</code>
<code>com.amazonaws.services.identitymanagement.AmazonIdentityManagementAsyncClient</code>	<code>software.amazon.awssdk.services.iam.IamAsyncClient</code>
<code>com.amazonaws.services.identitymanagement.AmazonIdentityManagementClient</code>	<code>software.amazon.awssdk.services.iam.IamClient</code>
<code>com.amazonaws.services.importexport.AmazonImportExportAsyncClient</code>	<code>software.amazon.awssdk.services.importexport.ImportExportAsyncClient</code>
<code>com.amazonaws.services.importexport.AmazonImportExportClient</code>	<code>software.amazon.awssdk.services.importexport.ImportExportClient</code>
<code>com.amazonaws.services.inspector.AmazonInspectorAsyncClient</code>	<code>software.amazon.awssdk.services.inspector.InspectorAsyncClient</code>
<code>com.amazonaws.services.inspector.AmazonInspectorClient</code>	<code>software.amazon.awssdk.services.inspector.InspectorClient</code>
<code>com.amazonaws.services.iot.AWSIoTAsyncClient</code>	<code>software.amazon.awssdk.services.iot.IotAsyncClient</code>
<code>com.amazonaws.services.iot.AWSIoTClient</code>	<code>software.amazon.awssdk.services.iot.IotClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesAsyncClient</code>	<code>software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesAsyncClient</code>
<code>com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesClient</code>	<code>software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesClient</code>
<code>com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsAsyncClient</code>	<code>software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsAsyncClient</code>
<code>com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsClient</code>	<code>software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsClient</code>
<code>com.amazonaws.services.iotanalytics.AWSIoTAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.iotanalytics.IotAnalyticsAsyncClient</code>
<code>com.amazonaws.services.iotanalytics.AWSIoTAnalyticsClient</code>	<code>software.amazon.awssdk.services.iotanalytics.IotAnalyticsClient</code>
<code>com.amazonaws.services.iotdata.AWSIoTDataAsyncClient</code>	<code>software.amazon.awssdk.services.iotdata.IotDataAsyncClient</code>
<code>com.amazonaws.services.iotdata.AWSIoTDataClient</code>	<code>software.amazon.awssdk.services.iotdata.IotDataClient</code>
<code>com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneAsyncClient</code>	<code>software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient</code>
<code>com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneClient</code>	<code>software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.kinesis.AmazonKinesisAsyncClient</code>	<code>software.amazon.awssdk.services.kinesis.KinesisAsyncClient</code>
<code>com.amazonaws.services.kinesis.AmazonKinesisClient</code>	<code>software.amazon.awssdk.services.kinesis.KinesisClient</code>
<code>com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsAsyncClient</code>
<code>com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsClient</code>	<code>software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsClient</code>
<code>com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseAsyncClient</code>	<code>software.amazon.awssdk.services.firehose.FirehoseAsyncClient</code>
<code>com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseClient</code>	<code>software.amazon.awssdk.services.firehose.FirehoseClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaClient</code>	<code>software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideo.KinesisVideoAsyncClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoClient</code>	<code>software.amazon.awssdk.services.kinesisvideo.KinesisVideoClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaClient</code>	<code>software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoPutMediaClient</code>	サポート外
<code>com.amazonaws.services.kms.AWSKMSAsyncClient</code>	<code>software.amazon.awssdk.services.kms.KmsAsyncClient</code>
<code>com.amazonaws.services.kms.AWSKMSClient</code>	<code>software.amazon.awssdk.services.kms.KmsClient</code>
<code>com.amazonaws.services.lambda.AWSLambdaAsyncClient</code>	<code>software.amazon.awssdk.services.lambda.LambdaAsyncClient</code>
<code>com.amazonaws.services.lambda.AWSLambdaClient</code>	<code>software.amazon.awssdk.services.lambda.LambdaClient</code>
<code>com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingAsyncClient</code>	<code>software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingAsyncClient</code>
<code>com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingClient</code>	<code>software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.lexruntime.AmazonLexRuntimeAsyncClient</code>	<code>software.amazon.awssdk.services.lexruntime.LexRuntimeAsyncClient</code>
<code>com.amazonaws.services.lexruntime.AmazonLexRuntimeClient</code>	<code>software.amazon.awssdk.services.lexruntime.LexRuntimeClient</code>
<code>com.amazonaws.services.lightsail.AmazonLightsailAsyncClient</code>	<code>software.amazon.awssdk.services.lightsail.LightsailAsyncClient</code>
<code>com.amazonaws.services.lightsail.AmazonLightsailClient</code>	<code>software.amazon.awssdk.services.lightsail.LightsailClient</code>
<code>com.amazonaws.services.logs.AWSLogsAsyncClient</code>	<code>software.amazon.awssdk.services.logs.LogsAsyncClient</code>
<code>com.amazonaws.services.logs.AWSLogsClient</code>	<code>software.amazon.awssdk.services.logs.LogsClient</code>
<code>com.amazonaws.services.machinelearning.AmazonMachineLearningAsyncClient</code>	<code>software.amazon.awssdk.services.machinelearning.MachineLearningAsyncClient</code>
<code>com.amazonaws.services.machinelearning.AmazonMachineLearningClient</code>	<code>software.amazon.awssdk.services.machinelearning.MachineLearningClient</code>
<code>com.amazonaws.services.macie.AmazonMacieAsyncClient</code>	<code>software.amazon.awssdk.services.macie.MacieAsyncClient</code>
<code>com.amazonaws.services.macie.AmazonMacieClient</code>	<code>software.amazon.awssdk.services.macie.MacieClient</code>



1.x クライアント	2.x クライアント
<code>com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsAsyncClient</code>
<code>com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsClient</code>	<code>software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsClient</code>
<code>com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementAsyncClient</code>	<code>software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementAsyncClient</code>
<code>com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementClient</code>	<code>software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementClient</code>
<code>com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringAsyncClient</code>	<code>software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringAsyncClient</code>
<code>com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringClient</code>	<code>software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringClient</code>
<code>com.amazonaws.services.mediaconvert.AWSMediaConvertAsyncClient</code>	<code>software.amazon.awssdk.services.mediaconvert.MediaConvertAsyncClient</code>
<code>com.amazonaws.services.mediaconvert.AWSMediaConvertClient</code>	<code>software.amazon.awssdk.services.mediaconvert.MediaConvertClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.medialive.AWSMediaLiveAsyncClient</code>	<code>software.amazon.awssdk.services.medialive.MediaLiveAsyncClient</code>
<code>com.amazonaws.services.medialive.AWSMediaLiveClient</code>	<code>software.amazon.awssdk.services.medialive.MediaLiveClient</code>
<code>com.amazonaws.services.mediapackage.AWSMediaPackageAsyncClient</code>	<code>software.amazon.awssdk.services.mediapackage.MediaPackageAsyncClient</code>
<code>com.amazonaws.services.mediapackage.AWSMediaPackageClient</code>	<code>software.amazon.awssdk.services.mediapackage.MediaPackageClient</code>
<code>com.amazonaws.services.mediastore.AWSMediaStoreAsyncClient</code>	<code>software.amazon.awssdk.services.mediastore.MediaStoreAsyncClient</code>
<code>com.amazonaws.services.mediastore.AWSMediaStoreClient</code>	<code>software.amazon.awssdk.services.mediastore.MediaStoreClient</code>
<code>com.amazonaws.services.mediastoredata.AWSMediaStoreDataAsyncClient</code>	<code>software.amazon.awssdk.services.mediastoredata.MediaStoreDataAsyncClient</code>
<code>com.amazonaws.services.mediastoredata.AWSMediaStoreDataClient</code>	<code>software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient</code>
<code>com.amazonaws.services.mediatailor.AWSMediaTailorAsyncClient</code>	<code>software.amazon.awssdk.services.mediatailor.MediaTailorAsyncClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.mediataylor.AWSMediaTailorClient</code>	<code>software.amazon.awssdk.services.mediataylor.MediaTailorClient</code>
<code>com.amazonaws.services.migrationhub.AWSMigrationHubAsyncClient</code>	<code>software.amazon.awssdk.services.migrationhub.MigrationHubAsyncClient</code>
<code>com.amazonaws.services.migrationhub.AWSMigrationHubClient</code>	<code>software.amazon.awssdk.services.migrationhub.MigrationHubClient</code>
<code>com.amazonaws.services.mobile.AWSMobileAsyncClient</code>	<code>software.amazon.awssdk.services.mobile.MobileAsyncClient</code>
<code>com.amazonaws.services.mobile.AWSMobileClient</code>	<code>software.amazon.awssdk.services.mobile.MobileClient</code>
<code>com.amazonaws.services.mq.AmazonMQAsyncClient</code>	<code>software.amazon.awssdk.services.mq.MqAsyncClient</code>
<code>com.amazonaws.services.mq.AmazonMQClient</code>	<code>software.amazon.awssdk.services.mq.MqClient</code>
<code>com.amazonaws.services.mturk.AmazonMTurkAsyncClient</code>	<code>software.amazon.awssdk.services.mturk.MTurkAsyncClient</code>
<code>com.amazonaws.services.mturk.AmazonMTurkClient</code>	<code>software.amazon.awssdk.services.mturk.MTurkClient</code>
<code>com.amazonaws.services.neptune.AmazonNeptuneAsyncClient</code>	<code>software.amazon.awssdk.services.neptune.NeptuneAsyncClient</code>
<code>com.amazonaws.services.neptune.AmazonNeptuneClient</code>	<code>software.amazon.awssdk.services.neptune.NeptuneClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.opsworks.AWSOpsWorksAsyncClient</code>	<code>software.amazon.awssdk.services.opsworks.OpsWorksAsyncClient</code>
<code>com.amazonaws.services.opsworks.AWSOpsWorksClient</code>	<code>software.amazon.awssdk.services.opsworks.OpsWorksClient</code>
<code>com.amazonaws.services.opsworkscm.AWSOpsWorksCMAsyncClient</code>	<code>software.amazon.awssdk.services.opsworkscm.OpsWorksCmAsyncClient</code>
<code>com.amazonaws.services.opsworkscm.AWSOpsWorksCMClient</code>	<code>software.amazon.awssdk.services.opsworkscm.OpsWorksCmClient</code>
<code>com.amazonaws.services.organizations.AWSOrganizationsAsyncClient</code>	<code>software.amazon.awssdk.services.organizations.OrganizationsAsyncClient</code>
<code>com.amazonaws.services.organizations.AWSOrganizationsClient</code>	<code>software.amazon.awssdk.services.organizations.OrganizationsClient</code>
<code>com.amazonaws.services.pi.AWSPIAsyncClient</code>	<code>software.amazon.awssdk.services.pi.PiAsyncClient</code>
<code>com.amazonaws.services.pi.AWSPIClient</code>	<code>software.amazon.awssdk.services.pi.PiClient</code>
<code>com.amazonaws.services.pinpoint.AmazonPinpointAsyncClient</code>	<code>software.amazon.awssdk.services.pinpoint.PinpointAsyncClient</code>
<code>com.amazonaws.services.pinpoint.AmazonPinpointClient</code>	<code>software.amazon.awssdk.services.pinpoint.PinpointClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.polly.AmazonPollyAsyncClient</code>	<code>software.amazon.awssdk.services.polly.PollyAsyncClient</code>
<code>com.amazonaws.services.polly.AmazonPollyClient</code>	<code>software.amazon.awssdk.services.polly.PollyClient</code>
<code>com.amazonaws.services.pricing.AWSPricingAsyncClient</code>	<code>software.amazon.awssdk.services.pricing.PricingAsyncClient</code>
<code>com.amazonaws.services.pricing.AWSPricingClient</code>	<code>software.amazon.awssdk.services.pricing.PricingClient</code>
<code>com.amazonaws.services.rds.AmazonRDSAsyncClient</code>	<code>software.amazon.awssdk.services.rds.RdsAsyncClient</code>
<code>com.amazonaws.services.rds.AmazonRDSClient</code>	<code>software.amazon.awssdk.services.rds.RdsClient</code>
<code>com.amazonaws.services.redshift.AmazonRedshiftAsyncClient</code>	<code>software.amazon.awssdk.services.redshift.RedshiftAsyncClient</code>
<code>com.amazonaws.services.redshift.AmazonRedshiftClient</code>	<code>software.amazon.awssdk.services.redshift.RedshiftClient</code>
<code>com.amazonaws.services.rekognition.AmazonRekognitionAsyncClient</code>	<code>software.amazon.awssdk.services.rekognition.RekognitionAsyncClient</code>
<code>com.amazonaws.services.rekognition.AmazonRekognitionClient</code>	<code>software.amazon.awssdk.services.rekognition.RekognitionClient</code>
<code>com.amazonaws.services.resourcegroups.AWSResourceGroupsAsyncClient</code>	<code>software.amazon.awssdk.services.resourcegroups.ResourceGroupsAsyncClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.resourcegroups.AWSResourceGroupsClient</code>	<code>software.amazon.awssdk.services.resourcegroups.ResourceGroupsClient</code>
<code>com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIAsyncClient</code>	<code>software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiAsyncClient</code>
<code>com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIClient</code>	<code>software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiClient</code>
<code>com.amazonaws.services.route53.AmazonRoute53AsyncClient</code>	<code>software.amazon.awssdk.services.route53.Route53AsyncClient</code>
<code>com.amazonaws.services.route53.AmazonRoute53Client</code>	<code>software.amazon.awssdk.services.route53.Route53Client</code>
<code>com.amazonaws.services.route53domains.AmazonRoute53DomainsAsyncClient</code>	<code>software.amazon.awssdk.services.route53domains.Route53DomainsAsyncClient</code>
<code>com.amazonaws.services.route53domains.AmazonRoute53DomainsClient</code>	<code>software.amazon.awssdk.services.route53domains.Route53DomainsClient</code>
<code>com.amazonaws.services.s3.AmazonS3Client</code>	<code>software.amazon.awssdk.services.s3.S3Client</code>
<code>com.amazonaws.services.sagemaker.AmazonSageMakerAsyncClient</code>	<code>software.amazon.awssdk.services.sagemaker.SageMakerAsyncClient</code>
<code>com.amazonaws.services.sagemaker.AmazonSageMakerClient</code>	<code>software.amazon.awssdk.services.sagemaker.SageMakerClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeAsyncClient</code>	<code>software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeAsyncClient</code>
<code>com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeClient</code>	<code>software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeClient</code>
<code>com.amazonaws.services.secretsmanager.AWSSecretsManagerAsyncClient</code>	<code>software.amazon.awssdk.services.secretsmanager.SecretsManagerAsyncClient</code>
<code>com.amazonaws.services.secretsmanager.AWSSecretsManagerClient</code>	<code>software.amazon.awssdk.services.secretsmanager.SecretsManagerClient</code>
<code>com.amazonaws.services.securitytoken.AWSSecurityTokenServiceAsyncClient</code>	<code>software.amazon.awssdk.services.sts.StsAsyncClient</code>
<code>com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClient</code>	<code>software.amazon.awssdk.services.sts.StsClient</code>
<code>com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryAsyncClient</code>	<code>software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryAsyncClient</code>
<code>com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryClient</code>	<code>software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.servermigration.AWSServerMigrationAsyncClient</code>	<code>software.amazon.awssdk.services.sms.SmsAsyncClient</code>
<code>com.amazonaws.services.servermigration.AWSServerMigrationClient</code>	<code>software.amazon.awssdk.services.sms.SmsClient</code>
<code>com.amazonaws.services.servicecatalog.AWSServiceCatalogAsyncClient</code>	<code>software.amazon.awssdk.services.servicecatalog.ServiceCatalogAsyncClient</code>
<code>com.amazonaws.services.servicecatalog.AWSServiceCatalogClient</code>	<code>software.amazon.awssdk.services.servicecatalog.ServiceCatalogClient</code>
<code>com.amazonaws.services.servicediscovery.AWSServiceDiscoveryAsyncClient</code>	<code>software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryAsyncClient</code>
<code>com.amazonaws.services.servicediscovery.AWSServiceDiscoveryClient</code>	<code>software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryClient</code>
<code>com.amazonaws.services.shield.AWSShieldAsyncClient</code>	<code>software.amazon.awssdk.services.shield.ShieldAsyncClient</code>
<code>com.amazonaws.services.shield.AWSShieldClient</code>	<code>software.amazon.awssdk.services.shield.ShieldClient</code>
<code>com.amazonaws.services.simplesdb.AmazonSimpleDBAsyncClient</code>	<code>software.amazon.awssdk.services.simplesdb.SimpleDbAsyncClient</code>
<code>com.amazonaws.services.simplesdb.AmazonSimpleDBClient</code>	<code>software.amazon.awssdk.services.simplesdb.SimpleDbClient</code>



1.x クライアント	2.x クライアント
<code>com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceAsyncClient</code>	<code>software.amazon.awssdk.services.ses.SesAsyncClient</code>
<code>com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClient</code>	<code>software.amazon.awssdk.services.ses.SesClient</code>
<code>com.amazonaws.services.simplesystemsmanagement.AWSSimpleSystemsManagementAsyncClient</code>	<code>software.amazon.awssdk.services.ssm.SsmAsyncClient</code>
<code>com.amazonaws.services.simplesystemsmanagement.AWSSimpleSystemsManagementClient</code>	<code>software.amazon.awssdk.services.ssm.SsmClient</code>
<code>com.amazonaws.services.simplesworkflow.AmazonSimpleWorkflowAsyncClient</code>	<code>software.amazon.awssdk.services.swf.SwfAsyncClient</code>
<code>com.amazonaws.services.simplesworkflow.AmazonSimpleWorkflowClient</code>	<code>software.amazon.awssdk.services.swf.SwfClient</code>
<code>com.amazonaws.services.snowball.AmazonSnowballAsyncClient</code>	<code>software.amazon.awssdk.services.snowball.SnowballAsyncClient</code>
<code>com.amazonaws.services.snowball.AmazonSnowballClient</code>	<code>software.amazon.awssdk.services.snowball.SnowballClient</code>
<code>com.amazonaws.services.sns.AmazonSNSAsyncClient</code>	<code>software.amazon.awssdk.services.sns.SnsAsyncClient</code>
<code>com.amazonaws.services.sns.AmazonSNSClient</code>	<code>software.amazon.awssdk.services.sns.SnsClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.sqs. AmazonSQSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.sqs.SqsAsyncClient</code>
<code>com.amazonaws.services.sqs. AmazonSQSClient</code>	<code>software.amazon.awssdk.serv ices.sqs.SqsClient</code>
<code>com.amazonaws.services.step functions.AWSStepFunctionsA syncClient</code>	<code>software.amazon.awssdk.serv ices.sfn.SfnAsyncClient</code>
<code>com.amazonaws.services.step functions.AWSStepFunctionsC lient</code>	<code>software.amazon.awssdk.serv ices.sfn.SfnClient</code>
<code>com.amazonaws.services.stor agegateway.AWSStorageGatewa yAsyncClient</code>	<code>software.amazon.awssdk.serv ices.storagegateway.Storage GatewayAsyncClient</code>
<code>com.amazonaws.services.stor agegateway.AWSStorageGatewa yClient</code>	<code>software.amazon.awssdk.serv ices.storagegateway.Storage GatewayClient</code>
<code>com.amazonaws.services.supp ort.AWSSupportAsyncClient</code>	<code>software.amazon.awssdk.serv ices.support.SupportAsyncClient</code>
<code>com.amazonaws.services.supp ort.AWSSupportClient</code>	<code>software.amazon.awssdk.serv ices.support.SupportClient</code>
<code>com.amazonaws.services.tran scribe.AmazonTranscribeAsyn cClient</code>	<code>software.amazon.awssdk.serv ices.transcribe.TranscribeA syncClient</code>
<code>com.amazonaws.services.tran scribe.AmazonTranscribeClient</code>	<code>software.amazon.awssdk.serv ices.transcribe.TranscribeC lient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.translate.AmazonTranslateAsyncClient</code>	<code>software.amazon.awssdk.services.translate.TranslateAsyncClient</code>
<code>com.amazonaws.services.translate.AmazonTranslateClient</code>	<code>software.amazon.awssdk.services.translate.TranslateClient</code>
<code>com.amazonaws.services.waf.AWSWAFAsyncClient</code>	<code>software.amazon.awssdk.services.waf.WafAsyncClient</code>
<code>com.amazonaws.services.waf.AWSWAFClient</code>	<code>software.amazon.awssdk.services.waf.WafClient</code>
<code>com.amazonaws.services.waf.AWSWAFRegionalAsyncClient</code>	<code>software.amazon.awssdk.services.waf.regional.WafRegionalAsyncClient</code>
<code>com.amazonaws.services.waf.AWSWAFRegionalClient</code>	<code>software.amazon.awssdk.services.waf.regional.WafRegionalClient</code>
<code>com.amazonaws.services.workdocs.AmazonWorkDocsAsyncClient</code>	<code>software.amazon.awssdk.services.workdocs.WorkDocsAsyncClient</code>
<code>com.amazonaws.services.workdocs.AmazonWorkDocsClient</code>	<code>software.amazon.awssdk.services.workdocs.WorkDocsClient</code>
<code>com.amazonaws.services.workmail.AmazonWorkMailAsyncClient</code>	<code>software.amazon.awssdk.services.workmail.WorkMailAsyncClient</code>
<code>com.amazonaws.services.workmail.AmazonWorkMailClient</code>	<code>software.amazon.awssdk.services.workmail.WorkMailClient</code>

1.x クライアント	2.x クライアント
<code>com.amazonaws.services.workspaces.AmazonWorkspacesAsyncClient</code>	<code>software.amazon.awssdk.services.workspaces.WorkSpacesAsyncClient</code>
<code>com.amazonaws.services.workspaces.AmazonWorkspacesClient</code>	<code>software.amazon.awssdk.services.workspaces.WorkSpacesClient</code>
<code>com.amazonaws.services.xray.AWSXRayAsyncClient</code>	<code>software.amazon.awssdk.services.xray.XRayAsyncClient</code>
<code>com.amazonaws.services.xray.AWSXRayClient</code>	<code>software.amazon.awssdk.services.xray.XRayClient</code>

## クライアント作成のデフォルト

バージョン 2.x では、デフォルトのクライアント作成ロジックに次の変更が加えられました。

- S3 のデフォルトの認証情報プロバイダーチェーンには、匿名認証情報が含まれなくなりました。を使用して S3 への匿名アクセスを手動で指定する必要があります `AnonymousCredentialsProvider`。
- デフォルトのクライアント作成に関連する以下の環境変数は異なります。

1.x	2.x
<code>AWS_CBOR_DISABLED</code>	<code>CBOR_ENABLED</code>
<code>AWS_ION_BINARY_DISABLE</code>	<code>BINARY_ION_ENABLED</code>

- デフォルトのクライアント作成に関連する次のシステムプロパティは異なります。

1.x	2.x
<code>com.amazonaws.sdk.disableEc2Metadata</code>	<code>aws.disableEc2Metadata</code>

1.x	2.x
<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	<code>aws.ec2MetadataServiceEndpoint</code>
<code>com.amazonaws.sdk.disableCbor</code>	<code>aws.cborEnabled</code>
<code>com.amazonaws.sdk.disableIoNBinary</code>	<code>aws.binaryIonEnabled</code>

- バージョン 2.x では、次のシステムプロパティはサポートされていません。

1.x

`com.amazonaws.sdk.disableCertChecking`

`com.amazonaws.sdk.enableDefaultMetrics`

`com.amazonaws.sdk.enableThrottledRetry`

`com.amazonaws.regions.RegionUtils.fileOverride`

`com.amazonaws.regions.RegionUtils.disableRemote`

`com.amazonaws.services.s3.disableImplicitGlobalClients`

`com.amazonaws.sdk.enableInRegionOptimizedMode`

- カスタムの `endpoints.json` ファイルからリージョンの設定をロードすることはサポートされなくなりました。

## クライアント設定

1.x では、クライアントまたはクライアントビルダーで `ClientConfiguration` インスタンスを設定して、SDK クライアント設定が変更されていました。バージョン 2.x では、クライアント設定は別々の設定クラスに分割されています。別の設定クラスを使用することで、同期と非同期クライアントで異なる HTTP クライアントを設定したとしても、同じ `ClientOverrideConfiguration` クラスを使用できます。

## Example バージョン 1.x のクライアント設定の場合

```
AmazonDynamoDBClientBuilder.standard()
    .withClientConfiguration(clientConfiguration)
    .build()
```

## Example バージョン 2.x の同期クライアント設定の場合

```
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 バージョン 2.x の非同期クライアント設定の場合

```
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 クライアント

### 注目すべき変更

- バージョン 2.x では、`ClientBuilder` を使用して実装を指定することで、実行時に使用する HTTP クライアントを変更できます `clientBuilder.httpClientBuilder`。
- `ClientBuilder` を使用して HTTP クライアントをサービスクライアントビルダー `clientBuilder.httpClient` に渡すと、サービスクライアントが閉じても HTTP クライアントはデフォルトで閉じられません。これにより、サービスクライアント間で HTTP クライアントを共有できます。
- 非同期 HTTP クライアントは、ノンブロッキング IO を使用するようになりました。
- 一部のオペレーションでは、パフォーマンスを向上させるために HTTP/2 を使用するようになりました。

### 設定の変更

設定	1.x	2.x Sync、Apache	2.x 非同期、Netty
	<pre>ClientConfiguration clientConfig =     new ClientConfiguration()</pre>	<pre>ApacheHttpClientBuilder httpClientBuilder =     ApacheHttpClientBuilder()</pre>	<pre>NettyNioAsyncHttpClientBuilder httpClientBuilder =     NettyNioAsyncHttpClientBuilder()</pre>
最大接続数	<pre>clientConfig.setMaxConnections(...) clientConfig.withMaxConnections(...)</pre>	<pre>httpClientBuilder.maxConnections(...)</pre>	<pre>httpClientBuilder.maxConcurrency(...)</pre>
接続タイムアウト	<pre>clientConfig.setConnectTimeout(...)</pre>	<pre>httpClientBuilder.connectTimeout(...)</pre>	<pre>httpClientBuilder.connectTimeout(...)</pre>

設定	1.x	2.x Sync、Apache	2.x 非同期、Netty
	<pre> connectionTimeout(...) clientConfig.withConnectionTimeout(...) </pre>	<pre> connectionTimeout(...) httpClientBuilder.connectionAcquisitionTimeout(...) </pre>	<pre> connectionTimeout(...) </pre>
ソケットタイムアウト	<pre> clientConfig.setSocketTimeout(...) clientConfig.withSocketTimeout(...) </pre>	<pre> httpClientBuilder.socketTimeout(...) </pre>	<pre> httpClientBuilder.writeTimeout(...) httpClientBuilder.readTimeout(...) </pre>
接続 TTL	<pre> clientConfig.setConnectionTTL(...) clientConfig.withConnectionTTL(...) </pre>	<pre> httpClientBuilder.connectionTimeToLive(...) </pre>	<pre> httpClientBuilder.connectionTimeToLive(...) </pre>
接続の最大アイドル	<pre> clientConfig.setConnectionMaxIdleMillis(...) clientConfig.withConnectionMaxIdleMillis(...) </pre>	<pre> httpClientBuilder.connectionMaxIdleTime(...) </pre>	<pre> httpClientBuilder.connectionMaxIdleTime(...) </pre>



設定	1.x	2.x Sync、Apache	2.x 非同期、Netty
非アクティブ後の検証	<pre>clientConfig.setValidateAfterInactivityMillis(...) clientConfig.withValidateAfterInactivityMillis(...)</pre>	サポートされていない ( <a href="#">リクエスト機能</a> )	サポートされていない ( <a href="#">リクエスト機能</a> )
ローカルアドレス	<pre>clientConfig.setLocalAddress(...) clientConfig.withLocalAddress(...)</pre>	<pre>httpClientBuilder.localAddress(...)</pre>	サポートされません
Expect-continue が有効	<pre>clientConfig.setUseExpectContinue(...) clientConfig.withUseExpectContinue(...)</pre>	<pre>httpClientBuilder.expectContinueEnabled(...)</pre>	サポートされていない ( <a href="#">リクエスト機能</a> )
接続リーパー	<pre>clientConfig.setUseReaper(...) clientConfig.withReaper(...)</pre>	<pre>httpClientBuilder.useIdleConnectionReaper(...)</pre>	<pre>httpClientBuilder.useIdleConnectionReaper(...)</pre>

設定	1.x	2.x Sync、Apache	2.x 非同期、Netty
	<pre>AmazonDynamoDBClientBuilder .standard() .withClientConfiguration( clientConfiguration) .build()</pre>	<pre>DynamoDbClient.builder() .httpClientBuilder( httpClientBuilder) .build()</pre>	<pre>DynamoDbAsyncClient.builder() .httpClientBuilder( httpClientBuilder) .build()</pre>

## HTTP クライアントプロキシ

設定	1.x	2.x Sync、Apache	2.x 非同期、Netty
	<pre>ClientConfiguration clientConfig = new ClientConfiguration()</pre>	<pre>ProxyConfiguration .Builder proxyConfig = ProxyConfiguration .builder()</pre>	<pre>ProxyConfiguration .Builder proxyConfig = ProxyConfiguration .builder()</pre>
プロキシのホスト	<pre>clientConfig.setProxyHost(...) clientConfig.withProxyHost(...)</pre>	<pre>proxyConfig.endpoint(...)</pre>	<pre>proxyConfig.host(...)</pre>
プロキシのポート	<pre>clientConfig.setProxyPort(...) clientConfig.withProxyPort(...)</pre>	<pre>proxyConfig.endpoint(...)</pre>	<pre>proxyConfig.port(...)</pre>

設定	1.x	2.x Sync、Apache	2.x 非同期、Netty
		<a href="#">プロキシポート</a> が埋め込まれている endpoint	
プロキシユーザー名	<pre>clientConfig.setProxyUsername(...) clientConfig.withProxyUsername(...)</pre>	<pre>proxyConfig.username(...)</pre>	<pre>proxyConfig.username(...)</pre>
プロキシのパスワード	<pre>clientConfig.setProxyPassword(...) clientConfig.withProxyPassword(...)</pre>	<pre>proxyConfig.password(...)</pre>	<pre>proxyConfig.password(...)</pre>
プロキシドメイン	<pre>clientConfig.setProxyDomain(...) clientConfig.withProxyDomain(...)</pre>	<pre>proxyConfig.ntlmDomain(...)</pre>	サポートされていない ( <a href="#">リクエスト機能</a> )
プロキシワークステーション	<pre>clientConfig.setProxyWorkspace(...) clientConfig.withProxyWorkstation(...)</pre>	<pre>proxyConfig.ntlmWorkstation(...)</pre>	サポートされていない ( <a href="#">リクエスト機能</a> )

設定	1.x	2.x Sync、Apache	2.x 非同期、Netty
プロキシ認証方法	<pre>clientConfig.setProxyAuthenticationMethods(...) clientConfig.withProxyAuthenticationMethods(...)</pre>	<u>サポートされていない</u>	サポートされていない ( <u>リクエスト機能</u> )
プリエンプティブ基本プロキシ認証	<pre>clientConfig.setPreemptiveBasicProxyAuth(...) clientConfig.withPreemptiveBasicProxyAuth(...)</pre>	<pre>proxyConfig.preemptiveBasicAuthenticationEnabled(...)</pre>	サポートされていない ( <u>リクエスト機能</u> )
非プロキシホスト	<pre>clientConfig.setNonProxyHosts(...) clientConfig.withNonProxyHosts(...)</pre>	<pre>proxyConfig.nonProxyHosts(...)</pre>	<pre>proxyConfig.nonProxyHosts(...)</pre>

設定	1.x	2.x Sync、Apache	2.x 非同期、Netty
ソケットプロキシを無効にする	<pre>clientConfig.setDisableSocketProxy(...) clientConfig.withDisableSocketProxy(...)</pre>	サポートされていない ( <a href="#">リクエスト機能</a> )	サポートされていない ( <a href="#">リクエスト機能</a> )
	<pre>AmazonDynamoDBClientBuilder     .standard()     .withClientConfiguration(clientConfiguration)     .build()</pre>	<pre>httpClientBuilder.proxyConfiguration(proxyConfiguration).build()</pre>	<pre>httpClientBuilder.proxyConfiguration(proxyConfiguration).build()</pre>

## クライアントオーバーライド

設定	1.x	2.x
	<pre>ClientConfiguration clientConfig =     new ClientConfiguration()</pre>	<pre>ClientOverrideConfiguration.Builder overrideConfig =     ClientOverrideConfiguration.builder()</pre>
ユーザーエージェントのプレフィックス	<pre>clientConfig.setUserAgentPrefix(...) clientConfig.withUserAgentPrefix(...)</pre>	<pre>overrideConfig.advancedOption(     SdkAdvancedClientOption.USER_AGENT_PREFIX, ...)</pre>

設定	1.x	2.x
ユーザーエージェントのサフィックス	<pre>clientConfig.setUserAgentSuffix(...) clientConfig.withUserAgentSuffix(...)</pre>	<pre>overrideConfig.advancedOption(     SdkAdvancedClientOption.USER_AGENT_SUFFIX, ...)</pre>
Signer	<pre>clientConfig.setSignerOverride(...) clientConfig.withSignerOverride(...)</pre>	<pre>overrideConfig.advancedOption(     SdkAdvancedClientOption.SIGNER, ...)</pre>
追加のヘッダー	<pre>clientConfig.addHeader(...) clientConfig.withHeader(...)</pre>	<pre>overrideConfig.putHeader(...)</pre>
リクエストのタイムアウト	<pre>clientConfig.setRequestTimeout(...) clientConfig.withRequestTimeout(...)</pre>	<pre>overrideConfig.apiCallAttemptTimeout(...)</pre>
クライアント実行タイムアウト	<pre>clientConfig.setClientExecutionTimeout(...) clientConfig.withClientExecutionTimeout(...)</pre>	<pre>overrideConfig.apiCallTimeout(...)</pre>
Gzip を使用する	<pre>clientConfig.setUseGzip(...) clientConfig.withGzip(...)</pre>	サポートされていない ( <a href="#">リクエスト機能</a> )

設定	1.x	2.x
ソケットバッファサイズヒント	<pre>clientConfig.setSocketBufferSizeHints(...) clientConfig.withSocketBufferSizeHints(...)</pre>	サポートされていない ( <a href="#">リクエスト機能</a> )
キャッシュレスポンスメタデータ	<pre>clientConfig.setCacheResponseMetadata(...) clientConfig.withCacheResponseMetadata(...)</pre>	サポートされていない ( <a href="#">リクエスト機能</a> )
レスポンスメタデータのキャッシュサイズ	<pre>clientConfig.setResponseMetadataCacheSize(...) clientConfig.withResponseMetadataCacheSize(...)</pre>	サポートされていない ( <a href="#">リクエスト機能</a> )
DNS リゾルバー	<pre>clientConfig.setDnsResolver(...) clientConfig.withDnsResolver(...)</pre>	サポートされていない ( <a href="#">リクエスト機能</a> )
TCP キープアライブ	<pre>clientConfig.setUseTcpKeepAlive(...) clientConfig.withTcpKeepAlive(...)</pre>	<p>このオプションは HTTP クライアント設定になりました</p> <ul style="list-style-type: none"> <li>- <code>ApacheHttpClient.builder().tcpKeepAlive(true)</code></li> <li>- <code>NettyNioAsyncHttpClient.builder().tcpKeepAlive(true)</code></li> </ul>

設定	1.x	2.x
安全なランダム	<pre>clientConfig.setSecureRandom(...) clientConfig.withSecureRandom(...)</pre>	サポートされていない ( <a href="#">リクエスト機能</a> )
	<pre>AmazonDynamoDBClientBuilder.standard()     .withClientConfiguration(clientConfiguration)     .build()</pre>	<pre>DynamoDbClient.builder()     .httpClientBuilder(httpClientBuilder)     .build()</pre>

## クライアントオーバーライドの再試行

設定	1.x	2.x
	<pre>ClientConfiguration clientConfig =     new ClientConfiguration()</pre>	<pre>RetryPolicy.Builder retryPolicy =     RetryPolicy.builder()</pre>
最大エラー再試行回数	<pre>clientConfig.setMaxErrorRetry(...) clientConfig.withMaxErrorRetry(...)</pre>	<pre>retryPolicy.numRetries(...)</pre>
スロットリングされた再試行を使用する	<pre>clientConfig.setUseThrottleRetries(...) clientConfig.withUseThrottleRetries(...)</pre>	サポートされません
スロットリング前の最大連続再試行回数	<pre>clientConfig.setMaxConsecutiveRetrie</pre>	サポートされません



設定	1.x	2.x
	<pre>sBeforeThrottling( ... ) clientConfig.withMaxCo nsecutiveRetriesBe foreThrottling(...)</pre>	
	<pre>AmazonDynamoDBClie ntBuilder.standard()     .withClientConfigu ration(clientConfi guration)     .build()</pre>	<pre>DynamoDbClient.bui lder()     .httpClientBuilder (httpClientBuilder)     .build()</pre>

## 非同期クライアント

設定	1.x	2.x
		<pre>ClientAsyncConfigu ration.Builder     asyncConfig =         ClientAsyncConfigu ration.builder()</pre>
エグゼキュター	<pre>AmazonDynamoDBAsyn cClientBuilder.sta ndard()     .withExecutorFacto ry(... )     .build()</pre>	<pre>asyncConfig.advanc edOption(     SdkAdvancedAsynCll ientOption.FUTURE_ COMPLETION_EXECUTO R, ...)</pre>
		<pre>DynamoDbAsynClien t.builder()     .asyncConfiguratio n(asyncConfig)     .build()</pre>

## その他のクライアントの変更

1.x からの次の ClientConfiguration オプションは、SDK の 2.x で変更されており、同等のものはありません。

設定	1.x	2.x 相当
[プロトコル]	<pre>clientConfig.setProtocol(Protocol.HTTP) clientConfig.withProtocol(Protocol.HTTP)</pre>	<p>プロトコル設定はデフォルトで HTTPS です。設定を変更するには、クライアントビルダーで HTTP エンドポイントを設定するプロトコルを指定します。</p> <pre>clientBuilder.endpointOverride(     URI.create("http://..."))</pre>

## 認証情報プロバイダーの変更

このセクションでは、AWS SDK for Java のバージョン 1.x と 2.x 間の認証情報プロバイダクラスとメソッドの名前変更のマッピングを提供しています。

### 顕著な違い

- デフォルトの認証情報プロバイダは、バージョン 2.x の環境変数前にシステムプロパティをロードします。詳細については、「[認証情報の使用](#)」を参照してください。
- コンストラクタメソッドは、create あるいは builder メソッドで置き換えられます。

#### Example

```
DefaultCredentialsProvider.create();
```

- 非同期リフレッシュはデフォルトで設定されなくなりました。認証情報プロバイダーの builder で指定する必要があります。

## Example

```
ContainerCredentialsProvider provider = ContainerCredentialsProvider.builder()
    .asyncCredentialUpdateEnabled(true)
    .build();
```

- `ProfileCredentialsProvider.builder()` を使用して、カスタムプロファイルファイルへのパスを指定できます。

## Example

```
ProfileCredentialsProvider profile = ProfileCredentialsProvider.builder()
    .profileFile(ProfileFile.builder().content(Paths.get("myProfileFile.file")).build())
    .build();
```

- プロファイルのファイル形式がより AWS CLI に一致するように変更されました。詳細については、「AWS Command Line Interface ユーザーガイド」の「[AWS CLI の設定](#)」を参照してください。

## バージョン 1.x と 2.x 間でマッピングされた認証情報プロバイダの変更

### AWSCredentialsProvider

カテゴリの変更	1.x	2.x
パッケージ/クラス名	<code>com.amazonaws.auth.AWSCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.AwsCredentialsProvider</code>
メソッド名	<code>getCredentials</code>	<code>resolveCredentials</code>
サポートされていないメソッド	<code>refresh</code>	非サポート

**DefaultAWSCredentialsProviderChain**

カテゴリの変更	1.x	2.x
パッケージ/クラス名	<code>com.amazonaws.auth.DefaultAWSCredentialsProviderChain</code>	<code>software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider</code>
作成	<code>new DefaultAWSCredentialsProviderChain</code>	<code>DefaultCredentialsProvider.create</code>
サポートされていないメソッド	<code>getInstance</code>	非サポート
外部設定の優先順位	システムプロパティの前の環境変数	環境変数の前のシステムプロパティ

**AWSStaticCredentialsProvider**

カテゴリの変更	1.x	2.x
パッケージ/クラス名	<code>com.amazonaws.auth.AWSStaticCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.StaticCredentialsProvider</code>
作成	<code>new AWSStaticCredentialsProvider</code>	<code>StaticCredentialsProvider.create</code>

**EnvironmentVariableCredentialsProvider**

カテゴリの変更	1.x	2.x
パッケージ/クラス名	<code>com.amazonaws.auth.EnvironmentVariable</code>	<code>software.amazon.awssdk.auth.credenti</code>

カテゴリの変更	1.x	2.x
	<code>leCredentialsProvider</code>	<code>als.EnvironmentVariableCredentialsProvider</code>
作成	<code>new EnvironmentVariableCredentialsProvider</code>	<code>EnvironmentVariableCredentialsProvider.create</code>
環境変数名	<code>AWS_ACCESS_KEY</code>	<code>AWS_ACCESS_KEY_ID</code>
	<code>AWS_SECRET_KEY</code>	<code>AWS_SECRET_ACCESS_KEY</code>

### SystemPropertiesCredentialsProvider

カテゴリの変更	1.x	2.x
パッケージ/クラス名	<code>com.amazonaws.auth.SystemPropertiesCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.SystemPropertyCredentialsProvider</code>
作成	<code>new SystemPropertiesCredentialsProvider</code>	<code>SystemPropertiesCredentialsProvider.create</code>
システムプロパティ名	<code>aws.secretKey</code>	<code>aws.secretAccessKey</code>

### ProfileCredentialsProvider

カテゴリの変更	1.x	2.x
パッケージ/クラス名	<code>com.amazonaws.auth.profile.ProfileCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials</code>

カテゴリの変更	1.x	2.x
		<code>als.ProfileCredentialsProvider</code>
作成	<code>new ProfileCredentialsProvider</code>	<code>ProfileCredentialsProvider.create</code>
カスタムプロファイルの場所	<ul style="list-style-type: none"> <li>• <code>AWS_CREDENTIAL_PROFILES_FILE</code> 環境変数</li> <li>• <code>new ProfileCredentialsProvider</code></li> </ul>	<ul style="list-style-type: none"> <li>• <code>AWS_SHARED_CREDENTIALS_FILE</code> 環境変数</li> <li>• <code>ProfileCredentialsProvider.builder</code></li> </ul>

### ContainerCredentialsProvider

カテゴリの変更	1.x	2.x
パッケージ/クラス名	<code>com.amazonaws.auth.ContainerCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider</code>
作成	<code>new ContainerCredentialsProvider</code>	<code>ContainerCredentialsProvider.create</code>
非同期更新を指定する	非サポート	デフォルトの動作

### InstanceProfileCredentialsProvider

カテゴリの変更	1.x	2.x
パッケージ/クラス名	<code>com.amazonaws.auth.InstanceProfileCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider</code>

カテゴリの変更	1.x	2.x
作成	<code>new InstanceProfileCredentialsProvider</code>	<code>InstanceProfileCredentialsProvider.create</code>
非同期更新を指定する	<code>new InstanceProfileCredentialsProvider(true)</code>	<code>InstanceProfileCredentialProvider.builder().asyncCredentialUpdateEnabled(true).build()</code>
システムプロパティ名	<code>com.amazonaws.sdk.disableEc2Metadata</code>	<code>aws.disableEc2Metadata</code>
	<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	<code>aws.ec2MetadataServiceEndpoint</code>

## STSAssumeRoleSessionCredentialsProvider

カテゴリの変更	1.x	2.x
パッケージ/クラス名	<code>com.amazonaws.auth.STSAssumeRoleSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsAssumeRoleCredentialsProvider</code>
作成	<ul style="list-style-type: none"> <li><code>new STSAssumeRoleSessionCredentialsProvider</code></li> <li><code>new STSAssumeRoleSessionCredentialsProvider.Builder</code></li> </ul>	<code>StsAssumeRoleCredentialsProvider.builder</code>
非同期更新	デフォルトの動作	デフォルトの動作

カテゴリの変更	1.x	2.x
構成	<code>new STSAssumeRoleSessionCredentialsProvider.Builder</code>	<code>StsClient</code> および <code>AssumeRoleRequest</code> リクエストを設定する

## STSSessionCredentialsProvider

カテゴリの変更	1.x	2.x
パッケージ/クラス名	<code>com.amazonaws.auth.STSSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsGetSessionTokenCredentialsProvider</code>
作成	<code>new STSAssumeRoleSessionCredentialsProvider</code>	<code>StsGetSessionTokenCredentialsProvider.builder</code>
非同期更新	デフォルトの動作	<code>StsGetSessionTokenCredentialsProvider.builder</code>
構成	コンストラクタパラメータ	ビルダーで <code>StsClient</code> および <code>GetSessionTokenRequest</code> リクエストを設定する

## WebIdentityFederationSessionCredentialsProvider

カテゴリの変更	1.x	2.x
パッケージ/クラス名	<code>com.amazonaws.auth.WebIdentityFederationSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsAssumeRoleWithWebIdentity</code>



カテゴリの変更	1.x	2.x
	<code>SessionCredentialsProvider</code>	<code>WithWebIdentityCredentialsProvider</code>
作成	<code>new WebIdentityFederationSessionCredentialsProvider</code>	<code>StsAssumeRoleWithWebIdentityCredentialsProvider.builder</code>
非同期更新	デフォルトの動作	<code>StsAssumeRoleWithWebIdentityCredentialsProvider.builder</code>
構成	コンストラクタパラメータ	ビルダーで <code>StsClient</code> および <code>AssumeRoleWithWebIdentityRequest</code> リクエストを設定する

## 置き換えられたクラス

1.x クラス	2.x 置換クラス
<code>com.amazonaws.auth.EC2ContainerCredentialsProviderWrapper</code>	<code>software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider</code> および <code>software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider</code>
<code>com.amazonaws.services.s3.S3CredentialsProviderChain</code>	<code>software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider</code> および <code>software.amazon.awssdk.auth.credentials.AnonymousCredentialsProvider</code>

## 削除されたクラス

### 1.x クラス

```
com.amazonaws.auth.ClasspathPropertiesFileCredentialsProvider
```

```
com.amazonaws.auth.PropertiesFileCredentialsProvider
```

## リージョンの変更

このセクションでは、Region と Regions クラスの使用に関する AWS SDK for Java 2.x における変更の導入を説明しています。

### リージョンの設定

- 一部の AWS サービスにはリージョン固有のエンドポイントがありません。これらのサービスを使用している場合、リージョンを Region.AWS\_GLOBAL または Region.AWS\_CN\_GLOBAL に設定する必要があります。

#### Example

```
Region region = Region.AWS_GLOBAL;
```

- com.amazonaws.regions.Regions と com.amazonaws.regions.Region クラスが 1 つのクラスである software.amazon.awssdk.regions.Region に結合されました。

## メソッドとクラス名のマッピング

次の表では、AWS SDK for Java の 1.x と 2.x 間のリージョン関連クラスをマッピングしています。上記のクラスのインスタンスを作成するには、of() メソッドを使用できます。

#### Example

```
RegionMetadata regionMetadata = RegionMetadata.of(Region.US_EAST_1);
```

## 1.x リージョンクラスメソッドの変更

1.x	2.x
<code>Regions.fromName</code>	<code>Region.of</code>
<code>Regions.getName</code>	<code>Region.id</code>
<code>Regions.getDescription</code>	<code>Region.metadata().description()</code>
<code>Regions.getCurrentRegion</code>	サポート外
<code>Regions.DEFAULT_REGION</code>	サポート外
<code>Regions.name</code>	<code>Region.id</code>

## 1.x リージョンクラスメソッドの変更

1.x	2.x
<code>Region.getName</code>	<code>Region.id</code>
<code>Region.hasHttpsEndpoint</code>	サポート外
<code>Region.hasHttpEndpoint</code>	サポート外
<code>Region.getAvailableEndpoints</code>	サポート外
<code>Region.createClient</code>	サポート外

## RegionMetadata クラスメソッドの変更

1.x	2.x
<code>RegionMetadata.getName</code>	<code>RegionMetadata.name</code>
<code>RegionMetadata.getDomain</code>	<code>RegionMetadata.domain</code>
<code>RegionMetadata.getPartition</code>	<code>RegionMetadata.partition</code>

## ServiceMetadata クラスメソッドの変更

1.x	2.x
<code>Region.getServiceEndpoint</code>	<code>ServiceMetadata.endpointFor(Region)</code>
<code>Region.isServiceSupported</code>	<code>ServiceMetadata.regions().contains(Region)</code>

## オペレーション、リクエスト、レスポンスの変更

SDK for Java の v2.x では、リクエストはクライアントオペレーションに渡されます。例えば、`DynamoDbClient'sPutItemRequest`は `DynamoDbClient.putItem`オペレーションに渡されます。これらのオペレーションは、AWS のサービスなどの からのレスポンスを返します `PutItemResponse`。

SDK for Java のバージョン 2.x には、1.x から次の変更があります。

- 複数のレスポンスページを含むオペレーションに、レスポンス内のすべての項目を自動的に反復処理 `Paginator` する方法が追加されました。
- リクエストとレスポンスは変更できません。
- リクエストとレスポンスは、コンストラクタの代わりに静的ビルダーメソッドを使用して作成する必要があります。例えば、1.x の `new PutItemRequest().withTableName(...)` は になりました `PutItemRequest.builder().tableName(...).build()`。
- オペレーションは、リクエストを作成するための簡単な方法をサポートしています：  
`dynamoDbClient.putItem(request -> request.tableName(...))`。

## ストリーミング操作

`Amazon S3` `getObject` や `putObject`メソッドなどのストリーミングオペレーションで、ノンブロッキング I/O がサポートされるようになりました。その結果、リクエストとレスポンス POJOs はパラメータ `InputStream` として受け取ることがなくなりました。代わりに、同期リクエストの場合 `RequestBody`、リクエストオブジェクトはバイトのストリームである を受け入れます。非同期同等物は を受け入れます `AsyncRequestBody`。

## Example 1.x での Amazon S3 **putObject** オペレーション

```
s3client.putObject(BUCKET, KEY, new File(file_path));
```

## Example 2.x での Amazon S3 **putObject** オペレーション

```
s3client.putObject(PutObjectRequest.builder()  
    .bucket(BUCKET)  
    .key(KEY)  
    .build(),  
    RequestBody.of(Paths.get("myfile.in")));
```

並行して、ストリーミングレスポンスオブジェクトは、同期クライアント `ResponseTransformer` の場合は `を`、非同期クライアント `AsyncResponseTransformer` の場合は `を受け入れます`。

## Example 1.x での Amazon S3 **getObject** オペレーション

```
S3Object o = s3.getObject(bucket, key);  
S3ObjectInputStream s3is = o.getObjectContent();  
FileOutputStream fos = new FileOutputStream(new File(key));
```

## Example 2.x での Amazon S3 **getObject** オペレーション

```
s3client.getObject(GetObjectRequest.builder().bucket(bucket).key(key).build(),  
    ResponseTransformer.toFile(Paths.get("key")));
```

SDK for Java 2.x では、ストリーミングレスポンスオペレーションには、レスポンスをメモリにロードし、一般的なインメモリタイプの変換を簡素化する `AsBytes` メソッドがあります。

## 例外変更

例外クラス名、構造、関係が変更されました。

`software.amazon.awssdk.core.exception.SdkException` は、他のすべての例外を拡張する新しい基本 `Exception` クラスです。

この表では、例外クラス名の変更をマッピングしています。

1.x	2.x
<code>com.amazonaws.SdkBaseException</code> <code>com.amazonaws.AmazonClientException</code>	<code>software.amazon.awssdk.core.exception.SdkException</code>
<code>com.amazonaws.SdkClientException</code>	<code>software.amazon.awssdk.core.exception.SdkClientException</code>
<code>com.amazonaws.AmazonServiceException</code>	<code>software.amazon.awssdk.awscore.exception.AwsServiceException</code>

以下の表では、バージョン 1.x と 2.x 間の例外クラスのメソッドをマッピングしています。

1.x	2.x
<code>AmazonServiceException.getRequestId</code>	<code>SdkServiceException.requestId</code>
<code>AmazonServiceException.getServiceName</code>	<code>AwsServiceException.awsErrorDetails().serviceName</code>
<code>AmazonServiceException.getErrorCode</code>	<code>AwsServiceException.awsErrorDetails().errorCode</code>
<code>AmazonServiceException.getErrorMessage</code>	<code>AwsServiceException.awsErrorDetails().errorMessage</code>
<code>AmazonServiceException.getStatusCode</code>	<code>AwsServiceException.awsErrorDetails().sdkHttpResponse().statusCode</code>
<code>AmazonServiceException.getHttpHeaders</code>	<code>AwsServiceException.awsErrorDetails().sdkHttpResponse().headers</code>

1.x	2.x
<code>AmazonServiceException.rawResponse</code>	<code>AwsServiceException.awsErrorResponseDetails().rawResponse</code>

## シリアル化の変更

SDK for Java v1.x と v2.x では、List オブジェクトをリクエストパラメータに対してシリアル化する方法が異なります。

SDK for Java 1.x は空のリストをシリアル化しませんが、SDK for Java 2.x は空のリストを空のパラメータとしてシリアル化します。

たとえば、`SampleRequest` を取得する `SampleOperation` を使用するサービスを考えてみましょう。`SampleRequest` は、次の例に示すように、String 型 `str1` と List 型 `listParam` の 2 つのパラメータを受け入れます。

### Example 1.x の `SampleOperation` の例

```
SampleRequest v1Request = new SampleRequest()
    .withStr1("TestName");

sampleServiceV1Client.sampleOperation(v1Request);
```

ワイヤレベルのロギングでは、`listParam` パラメータがシリアル化されていないことが示されています。

```
Action=SampleOperation&Version=2011-01-01&str1=TestName
```

### Example 2.x の `SampleOperation` の例

```
sampleServiceV2Client.sampleOperation(b -> b
    .str1("TestName"));
```

ワイヤレベルのロギングでは、値がないまま `listParam` パラメータがシリアル化されたことが示されています。

```
Action=SampleOperation&Version=2011-01-01&str1=TestName&listParam=
```

## サービス固有の変更

### Amazon S3 の変更

SDK for Java 2.x は、デフォルトで匿名アクセスを無効にします。そのため、`AnonymousCredentialsProvider` を使用して匿名アクセスを有効にする必要があります。

#### オペレーション名の変更

Amazon S3 クライアントの多くのオペレーション名は、AWS SDK for Java 2.x で変更されました。バージョン 1.x では、Amazon S3 クライアントはサービス API から直接生成されません。その結果、SDK オペレーションとサービス API 間で不整合が発生しました。バージョン 2.x では、サービス API に対してより互換性がある Amazon S3 クライアントを生成するようになりました。

次の表は、2 つのバージョンのオペレーション名を示しています。

#### Amazon S3 オペレーション名

1.x	2.x
<code>abortMultipartUpload</code>	<code>abortMultipartUpload</code>
<code>changeObjectStorageClass</code>	<code>copyObject</code>
<code>completeMultipartUpload</code>	<code>completeMultipartUpload</code>
<code>copyObject</code>	<code>copyObject</code>
<code>copyPart</code>	<code>uploadPartCopy</code>
<code>createBucket</code>	<code>createBucket</code>
<code>deleteBucket</code>	<code>deleteBucket</code>
<code>deleteBucketAnalyticsConfiguration</code>	<code>deleteBucketAnalyticsConfiguration</code>
<code>deleteBucketCrossOriginConfiguration</code>	<code>deleteBucketCors</code>
<code>deleteBucketEncryption</code>	<code>deleteBucketEncryption</code>



1.x	2.x
<code>deleteBucketInventoryConfiguration</code>	<code>deleteBucketInventoryConfiguration</code>
<code>deleteBucketLifecycleConfiguration</code>	<code>deleteBucketLifecycle</code>
<code>deleteBucketMetricsConfiguration</code>	<code>deleteBucketMetricsConfiguration</code>
<code>deleteBucketPolicy</code>	<code>deleteBucketPolicy</code>
<code>deleteBucketReplicationConfiguration</code>	<code>deleteBucketReplication</code>
<code>deleteBucketTaggingConfiguration</code>	<code>deleteBucketTagging</code>
<code>deleteBucketWebsiteConfiguration</code>	<code>deleteBucketWebsite</code>
<code>deleteObject</code>	<code>deleteObject</code>
<code>deleteObjectTagging</code>	<code>deleteObjectTagging</code>
<code>deleteObjects</code>	<code>deleteObjects</code>
<code>deleteVersion</code>	<code>deleteObject</code>
<code>disableRequesterPays</code>	<code>putBucketRequestPayment</code>
<code>doesBucketExist</code>	<code>headBucket</code>
<code>doesBucketExistV2</code>	<code>headBucket</code>
<code>doesObjectExist</code>	<code>headObject</code>
<code>enableRequesterPays</code>	<code>putBucketRequestPayment</code>
<code>generatePresignedUrl</code>	<a href="#">S3Presigner</a>
<code>getBucketAccelerateConfiguration</code>	<code>getBucketAccelerateConfiguration</code>

1.x	2.x
<code>getBucketAcl</code>	<code>getBucketAcl</code>
<code>getBucketAnalyticsConfiguration</code>	<code>getBucketAnalyticsConfiguration</code>
<code>getBucketCrossOriginConfiguration</code>	<code>getBucketCors</code>
<code>getBucketEncryption</code>	<code>getBucketEncryption</code>
<code>getBucketInventoryConfiguration</code>	<code>getBucketInventoryConfiguration</code>
<code>getBucketLifecycleConfiguration</code>	<code>getBucketLifecycle</code> または <code>getBucketLifecycleConfiguration</code>
<code>getBucketLocation</code>	<code>getBucketLocation</code>
<code>getBucketLoggingConfiguration</code>	<code>getBucketLogging</code>
<code>getBucketMetricsConfiguration</code>	<code>getBucketMetricsConfiguration</code>
<code>getBucketNotificationConfiguration</code>	<code>getBucketNotification</code> または <code>getBucketNotificationConfiguration</code>
<code>getBucketPolicy</code>	<code>getBucketPolicy</code>
<code>getBucketReplicationConfiguration</code>	<code>getBucketReplication</code>
<code>getBucketTaggingConfiguration</code>	<code>getBucketTagging</code>
<code>getBucketVersioningConfiguration</code>	<code>getBucketVersioning</code>
<code>getBucketWebsiteConfiguration</code>	<code>getBucketWebsite</code>
<code>getObject</code>	<code>getObject</code>
<code>getObjectAcl</code>	<code>getObjectAcl</code>

1.x	2.x
<code>getObjectAsString</code>	<code>getObjectAsBytes().asUtf8String</code>
<code>getObjectMetadata</code>	<code>headObject</code>
<code>getObjectTagging</code>	<code>getObjectTagging</code>
<code>getResourceUrl</code>	<a href="#">S3Utilities#getUrl</a>
<code>getS3AccountOwner</code>	<code>listBuckets</code>
<code>getUrl</code>	<a href="#">S3Utilities#getUrl</a>
<code>headBucket</code>	<code>headBucket</code>
<code>initiateMultipartUpload</code>	<code>createMultipartUpload</code>
<code>isRequesterPaysEnabled</code>	<code>getBucketRequestPayment</code>
<code>listBucketAnalyticsConfigurations</code>	<code>listBucketAnalyticsConfigurations</code>
<code>listBucketInventoryConfigurations</code>	<code>listBucketInventoryConfigurations</code>
<code>listBucketMetricsConfigurations</code>	<code>listBucketMetricsConfigurations</code>
<code>listBuckets</code>	<code>listBuckets</code>
<code>listMultipartUploads</code>	<code>listMultipartUploads</code>
<code>listNextBatchOfObjects</code>	<code>listObjectsV2Paginator</code>
<code>listNextBatchOfVersions</code>	<code>listObjectVersionsPaginator</code>
<code>listObjects</code>	<code>listObjects</code>
<code>listObjectsV2</code>	<code>listObjectsV2</code>
<code>listParts</code>	<code>listParts</code>

1.x	2.x
<code>listVersions</code>	<code>listObjectVersions</code>
<code>putObject</code>	<code>putObject</code>
<code>restoreObject</code>	<code>restoreObject</code>
<code>restoreObjectV2</code>	<code>restoreObject</code>
<code>selectObjectContent</code>	<code>selectObjectContent</code>
<code>setBucketAccelerateConfiguration</code>	<code>putBucketAccelerateConfiguration</code>
<code>setBucketAcl</code>	<code>putBucketAcl</code>
<code>setBucketAnalyticsConfiguration</code>	<code>putBucketAnalyticsConfiguration</code>
<code>setBucketCrossOriginConfiguration</code>	<code>putBucketCors</code>
<code>setBucketEncryption</code>	<code>putBucketEncryption</code>
<code>setBucketInventoryConfiguration</code>	<code>putBucketInventoryConfiguration</code>
<code>setBucketLifecycleConfiguration</code>	<code>putBucketLifecycle</code> または <code>putBucketLifecycleConfiguration</code>
<code>setBucketLoggingConfiguration</code>	<code>putBucketLogging</code>
<code>setBucketMetricsConfiguration</code>	<code>putBucketMetricsConfiguration</code>
<code>setBucketNotificationConfiguration</code>	<code>putBucketNotification</code> または <code>putBucketNotificationConfiguration</code>
<code>setBucketPolicy</code>	<code>putBucketPolicy</code>
<code>setBucketReplicationConfiguration</code>	<code>putBucketReplication</code>

1.x	2.x
setBucketTaggingConfiguration	putBucketTagging
setBucketVersioningConfiguration	putBucketVersioning
setBucketWebsiteConfiguration	putBucketWebsite
setObjectAcl	putObjectAcl
setObjectRedirectLocation	copyObject
setObjectTagging	putObjectTagging
uploadPart	uploadPart

## Amazon SNS の変更

SNS クライアントは、アクセスするように設定されているリージョン以外のリージョンの SNS トピックにアクセスできなくなります。

## Amazon SQS の変更

SQS クライアントは、アクセスするように設定されているリージョン以外のリージョンの SQS キューにアクセスできなくなります。

## Amazon RDS の変更

SDK for Java 2.x は、1.x `RdsIamAuthTokenGenerator` の `クラスRdsUtilities#generateAuthenticationToken` の代わりに `を使用します。`

## プロファイルファイルの変更

は AWS SDK for Java 2.x `~/ .aws/config` と `~/ .aws/credentials` し、CLI AWS がファイルを解析する方法をより厳密にエミュレートします。

SDK for Java 2.x:

- `~/` または `の~`後に、パスの先頭にあるファイルシステムのデフォルトパス区切り文字を解決します。このために、`、`、`$USERPROFILE` (Windows のみ)、`$HOME`、`$HOMEPATH` (Windows のみ)、`$HOMEDRIVE`、システム `user.home` プロパティの順にチェックします。

- の代わりに `AWS_SHARED_CREDENTIALS_FILE` 環境変数を探します  
す `AWS_CREDENTIAL_PROFILES_FILE`。
- プロファイル名の先頭に という単語なしで、設定ファイルのプロファイル定義をサイレント `profile` に削除します。
- 英数字、アンダースコア、ダッシュ文字 (設定ファイルの先頭の単語が削除された後) `profile` で構成されないプロファイル定義は、そのまま削除されます。
- 同じファイル内で複製されたプロファイル定義の設定をマージします。
- 設定ファイルと認証情報ファイルの両方で複製されたプロファイル定義の設定をマージします。
- `[profile foo]` と の両方が同じファイルにある場合 `[foo]`、設定をマージしません。
- `[profile foo]` と の両方 `[foo]` が設定ファイルで見つかった `[profile foo]` 場合は、 の設定を使用します。
- 同じファイルとプロファイル内の最後の重複した設定の値を使用します。
- コメントを定義する `#` ために、 `;` と の両方を認識します。
- 文字が閉じ括弧の横にある場合でも、プロファイル定義 `#` で `;` と を認識してコメントを定義します。
- `;` と を認識 `#` して、先頭に空白がある場合にのみ設定値にコメントを定義します。
- と `;`、`#` および以下のすべてのコンテンツの前に空白がない場合に、設定値を認識します。
- ロールベースの認証情報を最も優先度の高い認証情報と見なします。2.x SDK は、ユーザーが `role_arn` プロパティを指定した場合、常にロールベースの認証情報を使用します。
- セッションベースの認証情報を `second-highest-priority` 認証情報と見なします。ロールベースの認証情報が使用されておらず、ユーザーが `aws_access_key_id` プロパティと `aws_session_token` プロパティを指定している場合、2.x SDK は常にセッションベースの認証情報を使用します。
- ロールベースおよびセッションベースの認証情報が使用されておらず、ユーザーが `aws_access_key_id` プロパティを指定した場合は、基本的な認証情報を使用します。

## 環境変数とシステムプロパティの変更

1.x 環境変数	1.x システムプロパティ	2.x 環境変数	2.x システムプロパティ
AWS_ACCESS_KEY_ID AWS_ACCESS_KEY	aws.accessKeyId	AWS_ACCESS_KEY_ID	aws.accessKeyId
AWS_SECRET_KEY AWS_SECRET_ACCESS_KEY	aws.secretKey	AWS_SECRET_ACCESS_KEY	aws.secretAccessKey
AWS_SESSION_TOKEN	aws.sessionToken	AWS_SESSION_TOKEN	aws.sessionToken
AWS_REGION	aws.region	AWS_REGION	aws.region
AWS_CONFIG_FILE		AWS_CONFIG_FILE	aws.configFile
AWS_CREDENTIALS_PROFILE_FILE		AWS_SHARED_CREDENTIALS_FILE	aws.sharedCredentialsFile
AWS_PROFILE	aws.profile	AWS_PROFILE	aws.profile
AWS_EC2_METADATA_DISABLED	com.amazonaws.sdk.disableEc2Metadata	AWS_EC2_METADATA_DISABLED	aws.disableEc2Metadata
	com.amazonaws.sdk.ec2MetadataServiceEndpointOverride	AWS_EC2_METADATA_SERVICE_ENDPOINT	aws.ec2MetadataServiceEndpoint

1.x 環境変数	1.x システムプロパティ	2.x 環境変数	2.x システムプロパティ
AWS_CONTAINER_CREDENTIALS_RELATIVE_URI		AWS_CONTAINER_CREDENTIALS_RELATIVE_URI	aws.containerCredentialsPath
AWS_CONTAINER_CREDENTIALS_FULL_URI		AWS_CONTAINER_CREDENTIALS_FULL_URI	aws.containerCredentialsFullUri
AWS_CONTAINER_AUTHORIZATION_TOKEN		AWS_CONTAINER_AUTHORIZATION_TOKEN	aws.containerAuthorizationToken
AWS_CBOR_DISABLED	com.amazonaws.sdk.disableCbor	CBOR_ENABLED	aws.cborEnabled
AWS_ION_BINARY_DISABLE	com.amazonaws.sdk.disableIonBinary	BINARY_ION_ENABLED	aws.binaryIonEnabled
AWS_EXECUTION_ENV		AWS_EXECUTION_ENV	aws.executionEnvironment
	com.amazonaws.sdk.disableCertificateChecking	サポート対象外 ( <a href="#">リンク</a> <a href="#">エラスト機能</a> )	サポート対象外 ( <a href="#">リンク</a> <a href="#">エラスト機能</a> )



1.x 環境変数	1.x システムプロパティ	2.x 環境変数	2.x システムプロパティ
	<code>com.amazonaws.sdk.enableDefaultMetrics</code>	サポートされません	サポートされません
	<code>com.amazonaws.sdk.enableThrottledRetry</code>	サポートされません	サポートされません
	<code>com.amazonaws.regions.RegionUtils.fileOverride</code>	サポート対象外 ( <a href="#">リンク</a> <a href="#">エスト機能</a> )	サポート対象外 ( <a href="#">リンク</a> <a href="#">エスト機能</a> )
	<code>com.amazonaws.regions.RegionUtils.disableRemote</code>	サポート対象外 ( <a href="#">リンク</a> <a href="#">エスト機能</a> )	サポート対象外 ( <a href="#">リンク</a> <a href="#">エスト機能</a> )
	<code>com.amazonaws.services.s3.disableImplicitGlobalClients</code>	サポート対象外 ( <a href="#">リンク</a> <a href="#">エスト機能</a> )	サポート対象外 ( <a href="#">リンク</a> <a href="#">エスト機能</a> )
	<code>com.amazonaws.sdk.enableInRegionOptimizedMode</code>	サポート対象外 ( <a href="#">リンク</a> <a href="#">エスト機能</a> )	サポート対象外 ( <a href="#">リンク</a> <a href="#">エスト機能</a> )

## ウェーターのバージョン 1 からバージョン 2 への変更

このトピックでは、ウェーターの機能の変更をバージョン 1 (v1) からバージョン 2 (v2) に詳しく説明します。

次の表は、DynamoDB ウェーターの違いを具体的に示しています。他のサービスのウェーターも同じパターンに従います。

### 高レベル変更

ウェータークラスは、サービスと同じ Maven アーティファクトにあります。

変更	v1	v2
Maven の依存関係	<pre>&lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;       &lt;version&gt; 1.12.680<sup>1</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;     &lt;artifact Id&gt;dynamodb&lt;/artif actId&gt;   &lt;/dependency&gt;</pre>	<pre>&lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt;       &lt;version&gt; 2.25.10<sup>2</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;dynamodb&lt;/artif actId&gt;   &lt;/dependency&gt;</pre>

変更	v1	v2
	<code>&lt;/dependencies&gt;</code>	<code>&lt;/dependencies&gt;</code>
パッケージ名	<code>com.amazonaws.services.dynamodbv2.waiters</code>	<code>software.amazon.awssdk.services.dynamodb.waiters</code>
クラス名	<a href="#">AmazonDynamoDBWaiters</a>	<ul style="list-style-type: none"> <li>同期: <a href="#">DynamoDbWaiter</a></li> <li>非同期: <a href="#">DynamoDbAsyncWaiter</a></li> </ul>

<sup>1</sup> [最新バージョン](#)。 <sup>2</sup> [最新バージョン](#)。

## API の変更

変更	v1	v2
ウェイターを作成する	<pre>AmazonDynamoDB client     = AmazonDynamoDBClientBuilder         .standard().build(); AmazonDynamoDBWaiters     waiter = client.waiters();</pre>	<p>同期 :</p> <pre>DynamoDbClient client     = DynamoDbClient.create(); DynamoDbWaiter waiter     = client.waiter();</pre> <p>非同期 :</p> <pre>DynamoDbAsyncClient     asyncClient =         DynamoDbAsyncClient.create(); DynamoDbAsyncWaiter     waiter = asyncClient.waiter();</pre>
テーブルが存在するまで待つ	同期 :	同期 :

変更	v1	v2
	<pre>waiter.tableExists()     .run(new WaiterParameters&lt;&gt;(         new DescribeTableRequest(tableName)))</pre> <p>非同期 :</p> <pre>waiter.tableExists()     .runAsync(new WaiterParameters()         .withRequest(new DescribeTableRequest(tableName)),         new WaiterHandler() {             @Override             public void onSuccess(                 AmazonWebServiceRequest amazonWebServiceRequest) {                 System.out.println("Table creation succeeded");             }             @Override             public void onFailure(Exception e) {                 e.printStackTrace();             }         }).get();</pre>	<pre>WaiterResponse&lt;DescribeTableResponse&gt; waiterResponse =     waiter.waitForTableExists(         r -&gt; r.tableName("myTable")); waiterResponse.matched().response()     .ifPresent(System.out::println);</pre> <p>非同期 :</p> <pre>waiter.waitForTableExists(r -&gt;     r.tableName(tableName))     .whenComplete((r, t) -&gt; {         if (t != null) {             t.printStackTrace();         } else {             System.out.println("Table creation succeeded");         }     }).join();</pre>

## 設定変更

変更	v1	v2
ポーリング戦略 (最大試行回数と固定遅延)	<pre> MaxAttemptsRetryStrategy maxAttemptsRetryStrategy =     new MaxAttemptsRetryStrategy(10);  FixedDelayStrategy fixedDelayStrategy =     new FixedDelayStrategy(3);  PollingStrategy pollingStrategy =     new PollingStrategy(maxAttemptsRetryStrategy,         fixedDelayStrategy);  waiter.tableExists().run(     new WaiterParameters&lt;&gt;(         new DescribeTableRequest(tableName),         pollingStrategy); </pre>	<pre> FixedDelayBackoffStrategy fixedDelayBackoffStrategy =     FixedDelayBackoffStrategy.create(         Duration.ofSeconds(3));  waiter.waitForTableExists(r -&gt; r.tableName(),     c -&gt; c.maxAttempts(10)         .backoffStrategy(fixedDelayBackoffStrategy)); </pre>

## Amazon S3 Transfer Manager のバージョン 1 からバージョン 2 への変更

このトピックでは、Amazon S3 Transfer Manager のバージョン 1 (v1) からバージョン 2 (v2) への変更点について詳しく説明します。

## 高レベル変更

変更	v1	v2
Maven の依存関係	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro groupId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;       &lt;version&gt; 1.12.587<sup>1</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManagem ent&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro groupId&gt;     &lt;artifact Id&gt;aws-java-sdk-s3&lt;/ artifactId&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; </pre>	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt;       &lt;version&gt; 2.21.21<sup>2</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManagem ent&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifactId&gt;s3- transfer-manager&lt;/art ifactId&gt;     &lt;/dependency&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk.crt&lt;/groupId&gt;     &lt;artifact Id&gt;aws-crt&lt;/artifa ctId&gt;     &lt;version&gt; 0.28.7<sup>3</sup>&lt;/version&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; </pre>

変更	v1	v2
パッケージ名	com.amazonaws.serv ices.s3.transfer	software.amazon.aw ssdk.transfer.s3
Class name	<a href="#">TransferManager</a>	<a href="#">S3TransferManager</a>

<sup>1</sup> [最新バージョン](#)。 <sup>2</sup> [最新バージョン](#)。 <sup>3</sup> [最新バージョン](#)。

## 設定 API の変更

設定	v1	v2
(ビルダーを入手する)	<pre>TransferManagerBuilder tmBuilder =     TransferManagerBui lder.standard();</pre>	<pre>S3TransferManager. Builder tmBuilder =     S3TransferManager. builder();</pre>
S3 クライアント	<pre>tmBuilder.withS3Cl ient(...); tmBuilder.setS3C lient(...);</pre>	<pre>tmBuilder.s3Client (...);</pre>
エグゼキューター	<pre>tmBuilder.withExec utorFactory(...); tmBuilder.setExecu torFactory(...);</pre>	<pre>tmBuilder.executor (...);</pre>
スレッドプールをシャットダウンする	<pre>tmBuilder.withShut DownThreadPools(...); tmBuilder.setS hutdownThreadPools (...);</pre>	サポート外。指定されたエグゼキューターは、S3TransferManager が閉じられたときにシャットダウンされません。
最小アップロードパートサイズ	<pre>tmBuilder.withMini mumUploadPartSize( ...);</pre>	<pre>S3AsyncClient s3 =     S3AsyncClient.crtB uilder().</pre>

設定	v1	v2
	<pre>tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>minimumPartSizeInBytes(...) .build();  tmBuilder.s3Client(s3);</pre>
マルチパートアップロードのしきい値	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 =     S3AsyncClient.crtBuilder()         .thresholdInBytes(...)         .build();  tmBuilder.s3Client(s3);</pre>
最小コピーパーツサイズ	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 =     S3AsyncClient.crtBuilder()         .minimumPartSizeInBytes(...)         .build();  tmBuilder.s3Client(s3);</pre>
マルチパートコピーのしきい値	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 =     S3AsyncClient.crtBuilder()         .thresholdInBytes(...)         .build();  tmBuilder.s3Client(s3);</pre>



設定	v1	v2
並列ダウンロードを無効にする	<pre>tmBuilder.withDisableParallelDownloads(...); tmBuilder.setDisableParallelDownloads(...);</pre>	<p>標準 Java ベースの S3 クライアントを転送マネージャに渡して、並列ダウンロードを無効にします。</p> <pre>S3AsyncClient s3 =     S3AsyncClient.builder().build();  tmBuilder.s3Client(s3);</pre>
常にマルチパート md5 を計算する	<pre>tmBuilder.withAlwaysCalculateMultiPartMd5(...); tmBuilder.setAlwaysCalculateMultiPartMd5(...);</pre>	サポート外。

## 動作の変更

並列転送には AWS CRT ベースの S3 クライアントが必要です

SDK for Java 2.x では、自動並列転送機能 (マルチパートアップロード/ダウンロード) が [AWS CRT ベースの S3 クライアント](#) から利用できます。並列転送機能を有効にするには、パフォーマンスを最大化するために [AWS 共通ランタイム \(CRT\) ライブラリ](#) の依存関係を明示的に追加する必要があります。

AWS CRT ベースの S3 クライアントだけが使用せずに S3TransferManager 並列転送のパフォーマンスを最大化します。v2 S3TransferManager には、ファイルやディレクトリの転送を容易にする追加の APIs が用意されています。

S3TransferManager が並列転送を実行する機能は、S3TransferManager の開始方法と、AWS Common Runtime (CRT) ライブラリが依存関係として宣言されているかどうかによって異なります。

次の表は、AWS CRT が依存関係として宣言されているかどうかにかかわらず、S3TransferManager v2 の 3 つの初期化シナリオを示しています。

S3TransferManager v2 初期化アプローチ	AWS CRT は依存関係として宣言されていますか？	
	はい	いいえ
<p><b>S3AsyncClient</b> インスタンスを渡さずに <b>S3TransferManager</b> を初期化する</p> <p>静的作成メソッド:</p> <pre>S3TransferManager.create();</pre> <p>- または -</p> <p>ビルダーメソッド:</p> <pre>S3TransferManager.builder().build();</pre>	 自動並列転送が有効	 自動並列転送が無効
<p><code>.crt*</code> () ビルダーメソッドのいずれかで構築された <b>S3AsyncClient</b> インスタンスを渡す</p> <pre>S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder().build(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre> <p>- または -</p> <pre>S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre>	 自動並列転送が有効	 ランタイムエラー
<p>標準ビルダーメソッドのいずれかで構築された <b>S3AsyncClient</b> インスタンスを渡して、転送マネージャーが CRT を参照しないようにする</p>		

S3TransferManager v2 初期化アプローチ	AWS CRT は依存関係として宣言されていますか？	
<pre data-bbox="133 279 899 428">S3AsyncClient s3AsyncClient = S3AsyncClient.builder().build(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre> <p data-bbox="115 491 250 525">- または -</p> <pre data-bbox="133 588 899 737">S3AsyncClient s3AsyncClient = S3AsyncClient.create(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre>	自動並列転送が無効	自動並列転送が無効

## バイト範囲フェッチによる並列ダウンロード

自動並列転送機能が有効になっている場合、S3 Transfer Manager v2 は [バイト範囲フェッチ](#) を使用してオブジェクトの特定の部分を並列して取得します (マルチパートダウンロード)。v2 でのオブジェクトのダウンロード方法は、オブジェクトが最初にアップロードされた方法には依存しません。すべてのダウンロードで、高いスループットと同時実行性のメリットが得られます。

これとは対照的に、S3 Transfer Manager v1 では、オブジェクトが最初にどのようにアップロードされたかが問題になります。S3 Transfer Manager v1 は、パーツがアップロードされたのと同じ方法でオブジェクトの一部を取得します。オブジェクトが元々単一のオブジェクトとしてアップロードされた場合、S3 Transfer Manager v1 はサブリクエストを使用してダウンロードプロセスを高速化することはできません。

## 失敗動作

S3 Transfer Manager v1 では、いずれかのサブリクエストが失敗すると、ディレクトリ転送リクエストは失敗します。v1 とは異なり、S3 Transfer Manager v2 から返される Future は、一部のサブリクエストが失敗しても正常に完了します。

そのため、Future が正常に完了した場合でも、[CompletedDirectoryDownload.failedTransfers\(\)](#) メソッドまたは [CompletedDirectoryUpload.failedTransfers\(\)](#) メソッドを使用してレスポンスにエラーがないかどうかを確認する必要があります。

## EC2 メタデータユーティリティのバージョン 1 からバージョン 2 への変更

このトピックでは、SDK for Java Amazon Elastic Compute Cloud (EC2) メタデータユーティリティのバージョン 1 (v1) からバージョン 2 (v2) への変更について詳しく説明します。

### 高レベル変更

変更	v1	v2
Maven の依存関係	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;       &lt;version&gt; 1.12.587<sup>1</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;     &lt;artifact Id&gt;aws-java-sdk-co re&lt;/artifactId&gt;   &lt;/dependency&gt; &lt;/dependencies&gt; </pre>	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt;       &lt;version&gt; 2.21.21<sup>2</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;imds&lt;/artifactId&gt;   &lt;/dependency&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;apache-client<sup>3</sup>&lt;/ artifactId&gt; </pre>

変更	v1	v2
		<pre>&lt;/dependency&gt; &lt;/dependencies&gt;</pre>
パッケージ名	com.amazonaws.util	software.amazon.awssdk.imds
インスタンス化のアプローチ	<p>静的ユーティリティメソッドを使用し、インスタンス化を行わない。</p> <pre>String localHostName =     EC2Metada taUtils.getLocalHo stName();</pre>	<p>静的ファクトリメソッドを使用する。</p> <pre>Ec2MetadataClient client = Ec2Metada taClient.create();</pre> <p>または、ビルダーのアプローチを使用する。</p> <pre>Ec2MetadataClient client = Ec2Metada taClient.builder()     .endpointMode(Endp ointMode.IPV6)     .build();</pre>
クライアントのタイプ	同期専用ユーティリティメソッド: EC2MetadataUtils	同期: Ec2MetadataClient 非同期: Ec2MetadataAsyncClient

<sup>1</sup> [最新バージョン](#)。 <sup>2</sup> [最新バージョン](#)。

<sup>3</sup> v2 apache-client モジュールの宣言に注目してください。EC2 メタデータユーティリティの V2 では、同期メタデータクライアント用の SdkHttpClient インターフェイスまたは非同期メタデータクライアント用の SdkAsyncHttpClient インターフェイスの実装が必要です。[???](#) セクションには、使用できる HTTP クライアントのリストが表示されています。

## メタデータのリクエスト

v1 では、パラメータを受け付けない静的メソッドを使用して EC2 リソースのメタデータをリクエストします。これとは対照的に、v2 では EC2 リソースへのパスをパラメータとして指定する必要があります。次の表は、それぞれのアプローチの違いをまとめたものです。

v1	v2
<pre>String userMetaData = EC2MetadataUtils.getUserData();</pre>	<pre>Ec2MetadataClient client = Ec2MetadataClient.create(); Ec2MetadataResponse response =     client.get("/latest/user-data"); String userMetaData =     response.asString();</pre>

[インスタンスメタデータのカテゴリ](#)を参照して、メタデータをリクエストするために指定する必要のあるパスを見つけてください。

### Note

v2 でインスタンスメタデータクライアントを使用するときは、メタデータを取得するすべてのリクエストに同じクライアントを使用することを心がける必要があります。

## 動作の変更

### JSON データ

EC2 では、ローカルで実行されているインスタンスメタデータサービス (IMDS) が一部のメタデータを JSON 形式の文字列として返します。その一例が、[インスタンス ID ドキュメント](#)の動的メタデータです。

v1 API にはインスタンス ID メタデータごとに個別のメソッドが含まれていますが、v2 API は JSON 文字列を直接返します。JSON 文字列を処理するには、[ドキュメント API](#) を使用して応答を解析し、JSON 構造に移動します。

次の表は、v1 と v2 でインスタンス ID ドキュメントのメタデータを取得する方法を比較したものです。

ユースケース	v1	v2
リージョンを取得する	<pre>InstanceInfo instanceInfo =     EC2MetadataUtils.getInstanceInfo(); String region =     instanceInfo.getRegion();</pre>	<pre>Ec2MetadataResponse response =     client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String region =     instanceInfo.asMap().get("region").asString();</pre>
インスタンス ID を取得する	<pre>InstanceInfo instanceInfo =     EC2MetadataUtils.getInstanceInfo(); String instanceId =     instanceInfo.getInstanceId();</pre>	<pre>Ec2MetadataResponse response =     client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String instanceId =     instanceInfo.asMap().get("instanceId").asString();</pre>
インスタンスタイプを取得する	<pre>InstanceInfo instanceInfo =     EC2MetadataUtils.getInstanceInfo(); String instanceType =     instanceInfo.getInstanceType();</pre>	<pre>Ec2MetadataResponse response =     client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String instanceType =     instanceInfo.asMap</pre>

ユースケース	v1	v2
		<code>() .get("instanceType").asString();</code>

## エンドポイント解決の違い

次の表は、エンドポイントを IMDS に対して解決するために SDK がチェックする場所を示しています。場所は優先度の高い順にリストされています。

v1	v2
システムプロパティ: <code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	クライアントビルダーの設定方法: <code>endpoint(...)</code>
環境変数: <code>AWS_EC2_METADATA_SERVICE_ENDPOINT</code>	システムプロパティ: <code>aws.ec2MetadataServiceEndpoint</code>
デフォルト値: <code>http://169.254.169.254</code>	Config ファイル: <code>~.aws/config</code> 、 <code>ec2_metadata_service_endpoint</code> 設定
	解決済み <code>endpoint-mode</code> に関連する値
	デフォルト値: <code>http://169.254.169.254</code>

## v2 のエンドポイント解決

ビルダーを使用してエンドポイントを明示的に設定すると、そのエンドポイント値が他のすべての設定よりも優先されます。次のコードを実行すると、`aws.ec2MetadataServiceEndpoint` システムプロパティと設定ファイル `ec2_metadata_service_endpoint` の設定が存在しても無視されません。

```
Ec2MetadataClient client = Ec2MetadataClient
    .builder()
    .endpoint(URI.create("endpoint.to.use"))
    .build();
```



## エンドポイントモード

v2 では、エンドポイントモードを指定して、メタデータクライアントが IPv4 または IPv6 のデフォルトのエンドポイント値を使用するように設定できます。v1 ではエンドポイントモードを使用できません。IPv4 で使用されるデフォルト値は `http://169.254.169.254` で、IPv6 では `http://[fd00:ec2::254]` です。

次の表に、エンドポイントモードを設定するさまざまな方法を優先順位の高い順に示します。

		使用できる値
クライアントビルダーの設定 方法: <code>endpointMode(...)</code>	<pre>Ec2MetadataClient client = Ec2MetadataClient .builder() .endpointMode(EndpointMode.IPV4) .build();</pre>	<code>EndpointMode.IPV4</code> , <code>EndpointMode.IPV6</code>
システムプロパティ	<code>aws.ec2MetadataServiceEndpointMode</code>	IPv4、IPv6 (大文字と小文字は区別されません)
設定ファイル: <code>~/.aws/config</code>	<code>ec2_metadata_service_endpoint</code> の設定	IPv4、IPv6 (大文字と小文字は区別されません)
以前の方法では未指定	IPv4 が使用される	

### v2 での SDK による `endpoint` または `endpoint-mode` の解決方法

- SDK は、クライアントビルダーのコードで設定した値を使用し、外部設定は無視します。`endpoint` および `endpointMode` が両方ともクライアントビルダーで呼び出されると SDK は例外をスローするため、SDK はユーザーが使用するメソッドのエンドポイント値を使用します。
- コードで値を設定しない場合、SDK は外部設定を調べます。最初にシステムプロパティを検索し、次に設定ファイル内の設定を検索します。
  - SDK は最初にエンドポイント値をチェックします。値が見つかったら、その値が使用されます。
  - 値が見つからない場合は、SDK はエンドポイントモード設定を探します。

3. 最後に、SDK が外部設定を見つけられず、コードでメタデータクライアントを設定していない場合は、SDK は IPv4 の値の `http://169.254.169.254` を使用します。

## IMDSv2

Amazon EC2 では、インスタンスメタデータにアクセスするための 2 つのアプローチを定義しています。

- インスタンスメタデータサービスバージョン 1 (IMDSv1) – リクエスト/レスポンスアプローチ
- インスタンスメタデータサービスバージョン 2 (IMDSv2) – セッション指向アプローチ

次の表は、Java SDK が IMDS をどのように使用するかを比較したものです。

v1	v2
IMDSv2 がデフォルトで使用される	IMDSv2 を常に使用する
リクエストごとにセッショントークンの取得を試み、セッショントークンの取得に失敗した場合は IMDSv1 にフォールバックする	セッショントークンを内部キャッシュに保持し、複数のリクエストで再利用する

SDK for Java 2.x は IMDSv2 のみをサポートしており、IMDSv1 にはフォールバックしません。

## 設定の違い

以下の表にさまざまな設定オプション一覧表示します。

構成	v1	v2
再試行	設定不可	ビルダーメソッド <code>retryPolicy(...)</code> で設定可能
HTTP	接続タイムアウトは <code>AWS_METADATA_SERVICE_TIMEOUT</code> 環境変数で設定可能。デフォルト値は 1 秒です。	HTTP クライアントを <code>httpClient(...)</code> ビルダーメソッドに渡すことで設定可能。HTTP クライアント

構成	v1	v2
		のデフォルトの接続タイムアウトは 2 秒です。

## v2 の HTTP 設定の例

以下の例に、メタデータクライアントの設定方法を示します。この例では、接続タイムアウトを設定し、Apache HTTP クライアントを使用します。

```
SdkHttpClient httpClient = ApacheHttpClient.builder()
    .connectionTimeout(Duration.ofSeconds(1))
    .build();

Ec2MetadataClient imdsClient = Ec2MetadataClient.builder()
    .httpClient(httpClient)
    .build();
```

## Amazon CloudFront の事前署名をバージョン 1 からバージョン 2 に変更

このトピックでは、Amazon のバージョン 1 (v1) CloudFront からバージョン 2 (v2) への変更について詳しく説明します。

### 高レベル変更

変更	v1	v2
Maven の依存関係	<pre>&lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;       &lt;version&gt; <b>1.12.587</b><sup>1</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;</pre>	<pre>&lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt;       &lt;version&gt; <b>2.21.21</b><sup>2</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;</pre>

変更	v1	v2
	<pre>                 &lt;scope&gt;im port&lt;/scope&gt;                 &lt;/dependency&gt;             &lt;/dependencies&gt;         &lt;/dependencyManageme nt&gt;         &lt;dependencies&gt;             &lt;dependency&gt;                 &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;                 &lt;artifact Id&gt;cloudfront&lt;/art ifactId&gt;                 &lt;/dependency&gt;             &lt;/dependencies&gt; </pre>	<pre>                 &lt;scope&gt;im port&lt;/scope&gt;                 &lt;/dependency&gt;             &lt;/dependencies&gt;         &lt;/dependencyManageme nt&gt;         &lt;dependencies&gt;             &lt;dependency&gt;                 &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;                 &lt;artifact Id&gt;cloudfront&lt;/art ifactId&gt;                 &lt;/dependency&gt;             &lt;/dependencies&gt; </pre>
パッケージ名	com.amazonaws.serv ices.cloudfront	software.amazon.aw ssdk.services.clou dfront
クラス名	<a href="#">CloudFrontUrlSigner</a>  <a href="#">CloudFrontCookieSigner</a>	<a href="#">CloudFrontUtilities</a>  <a href="#">SignedUrl</a>  <a href="#">CannedSignerRequest</a>  <a href="#">CustomSignerRequest</a>

<sup>1</sup> [最新バージョン](#)。 <sup>2</sup> [最新バージョン](#)。

## API の変更

Behavior	v1	v2
既定リクエストを構築する	引数は API に直接渡されま す。	<pre> CannedSignerRequest cannedRequest =     CannedSig nerRequest.builder() </pre>

Behavior	v1	v2
		<pre>.resourceUrl(resourceUrl)  .privateKey(privateKey)  .keyPairId(keyPairId)  .expirationDate(expirationDate)  .build();</pre>
カスタムリクエストを構築する	引数は API に直接渡されます。	<pre>CustomSignerRequest customRequest =     CustomSignerRequest.builder()          .resourceUrl(resourceUrl)          .privateKey(keyFile)          .keyPairId(keyPairId)          .expirationDate(expirationDate)          .activeDate(activeDate)          .ipRange(ipRange)          .build();</pre>

Behavior	v1	v2
署名付き URL の生成 (既定)	<pre>String signedUrl =     CloudFrontUrlSigner.getSignedURLWith     CannedPolicy(         resourceUrl,         keyPairId, privateKe         y, expirationDate);</pre>	<pre>CloudFrontUtilities     cloudFrontUtilities =         CloudFrontUtilitie         s.create();  SignedUrl signedUrl =          cloudFrontUtilitie         s.getSignedUrlWith         CannedPolicy(canne         dRequest);  String url = signedUrl         .url();</pre>
署名付き Cookie の生成 (カスタム)	<pre>CookiesForCustomPolicy     cookies =         CloudFrontCookieSi         gner.getCookiesFor         CustomPolicy(             resourceUrl,             privateKey, keyPairId             , expirationDate,                 activeDate,             ipRange);</pre>	<pre>CloudFrontUtilities     cloudFrontUtilities =         CloudFrontUtilitie         s.create();  CookiesForCustomPolicy     cookies =         cloudFrontUtilitie         s.getCookiesForCus         tomPolicy(customRe         quest);</pre>

### v2 でリファクタリングされた Cookie ヘッダー

Java v1 では、Java SDK は Cookie ヘッダーを `Map.Entry<String, String>` として配信します。

```
Map.Entry<String, String> signatureMap = cookies.getSignature();
String signatureKey = signatureMap.getKey(); // "CloudFront-Signature"
String signatureValue = signatureMap.getValue(); // "[SIGNATURE_VALUE]"
```

Java v2 SDK は、ヘッダー全体を単一の `String` として配信します。

```
String signatureHeaderValue = cookies.signatureHeaderValue(); // "CloudFront-
Signature=[SIGNATURE_VALUE]"
```

## Amazon S3 URIs の解析をバージョン 1 からバージョン 2 に変更

このトピックでは、Amazon S3 URIs をバージョン 1 (v1) からバージョン 2 (v2) に解析する際の変更について詳しく説明します。

### 高レベル変更

v1 で S3 URI の解析を開始するには、コンストラクター `AmazonS3URI` を使用して をインスタンス化します。v2 では、 のインスタンス `parseUri()` で を呼び出して `S3Utilities`、 を返します `S3URI`。

変更	v1	v2
Maven の依存関係	<pre>&lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;       &lt;version&gt; <b>1.12.587</b><sup>1</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;</pre>	<pre>&lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt;       &lt;version&gt; <b>2.21.21</b><sup>2</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;s3&lt;/artifactId&gt;</pre>

変更	v1	v2
	<pre>&lt;artifact Id&gt;s3&lt;/artifactId&gt; &lt;/dependency&gt; &lt;/dependencies&gt;</pre>	<pre>&lt;/dependency&gt; &lt;/dependencies&gt;</pre>
パッケージ名	com.amazonaws.serv ices.s3	software.amazon.aw ssdk.services.s3
クラス名	<a href="#">AmazonS3URI</a>	<a href="#">S3URI</a>

<sup>1</sup> [最新バージョン](#)。 <sup>2</sup> [最新バージョン](#)。

## API の変更

Behavior	v1	v2
S3 URI を解析します。	<pre>URI uri = URI.creat e( "https://s3.amazon aws.com");  AmazonS3Uri s3Uri =     new AmazonS3U RI(uri, false);</pre>	<pre>S3Client s3Client =     S3Client.create(); S3Utilities s3Utiliti es =     s3Client.utilities ();  S3Uri s3Uri =     s3Utilities.parseU ri(uri);</pre>
S3 URI からバケット名を取得 します。	<pre>String bucket = s3Uri.getBucket();</pre>	<pre>Optional&lt;String&gt; bucket = s3Uri.bucket();</pre>
キーを取得します。	<pre>String key = s3Uri.get Key();</pre>	<pre>Optional&lt;String&gt; key = s3Uri.key();</pre>
リージョンを取得します。	<pre>String region = s3Uri.getRegion();</pre>	<pre>Optional&lt;Region&gt; region = s3Uri.region();</pre>



Behavior	v1	v2
		<pre>String region; if (s3Uri.region().isPresent()) {     region = s3Uri.region().get().id(); }</pre>
S3 URI がパススタイルであるかどうかを取得します。	<pre>boolean isPathStyle = s3Uri.isPathStyle();</pre>	<pre>boolean isPathStyle = s3Uri.isPathStyle();</pre>
バージョン ID を取得します。	<pre>String versionId = s3Uri.getVersionId();</pre>	<pre>Optional&lt;String&gt; versionId = s3Uri.firstMatchingRawQueryParameter( "versionId");</pre>
クエリパラメータを取得します。	該当なし	<pre>Map&lt;String, List&lt;String&gt;&gt; queryParams = s3Uri.rawQueryParameters();</pre>

## 動作の変更

### URL エンコーディング

v1 には、URI を URL エンコードするかどうかを指定するフラグを渡すオプションがあります。デフォルト値は、true です。

v2 では、URL エンコーディングはサポートされていません。予約文字または安全でない文字を含むオブジェクトキーまたはクエリパラメータを使用する場合は、それらを URL エンコードする必要があります。例えば、空白を " " に置き換える必要があります%20。

## IAM Policy Builder API のバージョン 1 からバージョン 2 への変更

このトピックでは、IAM Policy Builder API のバージョン 1 (v1) からバージョン 2 (v2) への変更について詳しく説明します。

## 高レベル変更

変更	v1	v2
Maven の依存関係	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;       &lt;version&gt; 1.12.587&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;     &lt;artifact Id&gt;aws-java-sdk-co re&lt;/artifactId&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; </pre>	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt;       &lt;version&gt; 2.21.21&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;iam-policy-buil der&lt;/artifactId&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; </pre>
パッケージ名	com.amazonaws.auth.policy	software.amazon.awssdk.policybuilder.iam
クラス名	<a href="#">Policy</a> <a href="#">Statement</a>	<a href="#">IamPolicy</a> <a href="#">IamStatement</a>

変更	v1	v2
	<ul style="list-style-type: none"> <li>• <a href="#">Statement.Effect</a></li> <li>• <a href="#">IdentityManagementActions</a></li> <li>• <a href="#">[リソース]</a></li> <li>• <a href="#">プリンシパル</a></li> <li>• <a href="#">条件</a></li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">IamEffect</a></li> <li>• <a href="#">IamAction</a></li> <li>• <a href="#">IamResource</a></li> <li>• <a href="#">IamPrincipal</a></li> <li>• <a href="#">IamCondition</a></li> <li>• <a href="#">IamConditionOperator</a></li> <li>• <a href="#">IamConditionKey</a></li> </ul>

<sup>1</sup> [最新バージョン](#)。 <sup>2</sup> [最新バージョン](#)。

## API の変更

設定	v1	v2
ポリシーをインスタンス化する	<pre>Policy policy = new Policy();</pre>	<pre>IamPolicy.Builder policyBuilder = IamPolicy.builder(); ... IamPolicy policy = policyBuilder.build();</pre>
ID を設定する	<pre>policy.withId(...); policy.setId(...);</pre>	<pre>policyBuilder.id(...);</pre>
バージョンの設定	該当なし - のデフォルトバージョンを使用します 2012-10-17	<pre>policyBuilder.version(...);</pre>
ステートメントの作成	<pre>Statement statement = new Statement (Effect.Allow) .withActions(...)</pre>	<pre>IamStatement statement = IamStatement.builder()</pre>

設定	v1	v2
	<pre> .withConditions(...) .withId(...) .withPrincipals(...) .withResources(...); </pre>	<pre> .effect(IamEffect.ALLOW) .actions(...) .notActions(...) .conditions(...) .sid(...) .principals(...) .notPrincipals(...) .resources(...) .notResources(...) .build() </pre>
Set ステートメント	<pre> policy.withStatements(statement); policy.setStatements(statement); </pre>	<pre> policyBuilder.addStatement(statement); </pre>

## ステートメントを作成する際の違い

### アクション

#### v1

v1 SDK には、ポリシーステートメントの [Action](#) 要素を表すサービスアクション [enum のタイプ](#) があります。以下の enum タイプがいくつかの例です。

- [IdentityManagementActions](#)
- [DynamoDBv2Actions](#)
- [SQSActions](#)

次の例は、のSendMessage定数を示していますSQSActions。

```
Action action = SQSActions.SendMessage;
```

v1 のステートメントに [NotAction](#)要素を指定することはできません。

v2

v2 では、[IamAction](#)インターフェイスはすべてのアクションを表します。[サービス固有のアクション](#)要素を指定するには、次のコードに示すように文字列を createメソッドに渡します。

```
IamAction action = IamAction.create("sqs:SendMessage");
```

次のコードに示すように、v2 でステートメント[NotAction](#)に を指定できます。

```
IamAction action = IamAction.create("sqs:SendMessage");  
IamStatement.builder().addNotAction(action);
```

条件

v1

ステートメントの条件を表すために、v1 SDK は のサブクラスを使用します[Condition](#)。

- [ArnCondition](#)
- [BooleanCondition](#)
- [DateCondition](#)
- [IpAddressCondition](#)
- [NumericCondition](#)
- [StringCondition](#)

各Conditionサブクラスは、条件の定義に役立つ比較enumタイプを定義します。例えば、次は条件の[文字列比較](#)とは異なります。

```
Condition condition = new StringCondition(StringComparisonType.StringNotLike, "key",  
"value");
```

## v2

v2 では、`IamConditionOperator` を使用してポリシーステートメントの条件を構築し [IamConditionOperator](#)、すべてのタイプenumsに を含む [IamCondition](#)を指定します。

```
IamCondition condition = IamCondition.create(IamConditionOperator.STRING_NOT_LIKE,
"key", "value");
```

## リソース

## v1

ポリシーステートメントの [Resource](#)要素は SDK の [Resource](#) クラスによって表されます。ARN は、コンストラクターで文字列として指定します。次のサブクラスはコンストラクターを提供します。

- [S3BucketResource](#)
- [S3ObjectResource](#)
- [SQSQueueResource](#)

v1 では、次のステートメントに示すように `withIsNotType`メソッドを呼び出す [Resource](#) ことで、の [NotResource](#)要素を指定できます。

```
Resource resource = new Resource("arn:aws:s3:::mybucket").withIsNotType(true);
```

## v2

v2 では、ARN を `IamResource.create`メソッドに渡すことで [Resource](#)要素を作成します。

```
IamResource resource = IamResource.create("arn:aws:s3:::mybucket");
```

次のスニペットに示すように [IamResource](#)、を [NotResource](#)要素として設定できます。

```
IamResource resource = IamResource.create("arn:aws:s3:::mybucket");
IamStatement.builder().addNotResource(resource);
```

`IamResource.ALL` はすべてのリソースを表します。

## プリンシパル

### v1

v1 SDK には、すべてのメンバーを含むプリンシパルのタイプを表す次の [Principal](#) クラスが用意されています。

- AllUsers
- AllServices
- AllWebProviders
- All

ステートメントに [NotPrincipal](#) 要素を追加することはできません。

### v2

v2 では、はすべてのプリンシパル `IamPrincipal.ALL` を表します。

他のタイプのプリンシパルのすべてのメンバーを表すには、を作成するときに [IamPrincipalType](#) クラスを使用します `IamPrincipal`。

- `IamPrincipal.create(IamPrincipalType.AWS, "*")` すべてのユーザーの。
- `IamPrincipal.create(IamPrincipalType.SERVICE, "*")` すべてのサービス用の。
- `IamPrincipal.create(IamPrincipalType.FEDERATED, "*")` すべてのウェブプロバイダー用の。
- `IamPrincipal.create(IamPrincipalType.CANONICAL_USER, "*")` すべての正規ユーザー用の。

次のステートメントに示すように、ポリシーステートメントを作成するときに、`addNotPrincipal` メソッドを使用して [NotPrincipal](#) 要素を表すことができます。

```
IamPrincipal principal = IamPrincipal.create(IamPrincipalType.AWS,
    "arn:aws:iam::444455556666:root");
IamStatement.builder().addNotPrincipal(principal);
```

## SDK for Java 1.x と 2.x を並行して使用する

プロジェクトで AWS SDK for Java の両方のバージョンを使用できます。

バージョン 1.x から Amazon S3 を使用し、バージョン 2.16.1 から DynamoDB を使用するプロジェクト用の pom.xml ファイルの例を以下に示します。

### Example POM の例

この例では、SDK の 1.x と 2.x バージョンの両方を使用するプロジェクトの pom.xml ファイルエントリを示しています。

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId>
      <version>1.12.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.16.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-s3</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>dynamodb</artifactId>
  </dependency>
</dependencies>
```



# AWS SDK for Java 用の OpenPGP 鍵

AWS SDK for Java 用に一般公開されている Maven アーティファクトはすべて OpenPGP 標準を使用して署名されています。アーティファクトの署名を検証するのに必要な公開鍵については、以下のセクションを参照してください。

## 現在のキー

次の表は、SDK for Java 1x と SDK for Java 2.x の最新リリースの OpenPGP 鍵情報を示しています。

キー ID	0xAC107B386692DADD
タイプ	RSA
サイズ	4096/4096
作成	2016-06-30
有効期限	2024-10-08
ユーザー ID	AWS SDK とツール <aws-dr-tools@amazon.com>
キーフィンガープリント	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 DADD

次の SDK for Java 用の OpenPGP 公開鍵をクリップボードにコピーするには、右上隅にある [コピー] アイコンを選択します。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
xsFNBFd1gAUBEACqbmFbxdJgz1lD7wrlskQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz  
mzJuQ+Kfjne2t+xTDex6MPJlMYp0viSWsX2psgvdmeyUpW9ap01rThNYkc+W5fRc  
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/  
KduGtZASqNAYLKR0mRODzUuaokLPo24pfm9bnr1RnRtw5ktPAA5bM9ZZaGKriej  
kT2lPffBbjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV  
u6PewUe2WP1nxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9b0ie
```

```
nj3QxLuQ1ZUKF79ES6JaM4t0z1gGcQeU1+Uk1gjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+ff+Xf0Cl6by0JFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFEMAuzzd0QJ4uz4xFFvaS0SZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVB0HKrq+fluorWepQivctzN6Y1N0kx5naTPGGaKWK7G2q
TbcY5SMnkIwFLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zsFNBFd1gAUBEAC8zNArPwb3dPMThL2xAY+fS60vXdB1Sk0tYJpDWpFgvo0d+VQ+
hV6Xu1GAHAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go
7xHIxgFjC046/06Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FK
VYR/j9uenEC/2NBcLuFy3q6cDfmCoDE0062kXMnaGz3knzEK/X1SkcjsxRDq7zaQ
lQ1Kou+3dICwy4x5SJQ8j1+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjy
pUwgp0MTo25gWxkvJlSJKU0b6b1786WnySIzF2gxqlkkEmB14RAssQkeXjrSmGws
MDyHNqyJeYFus18sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6Qxa
Zje9YSZUijGmZIdEBleRVt3Svhi8MY1nasd4bW2RK1sr7plkBf8QRe6biiQRf3KD
0Sn5CbmXpAcHJ1ZHRRdkXZDNQC6vCJxsy1300TrhJtAV1Yq347uyUbVi291ISVg
roUVtprismHoEk5Go0THbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbtnbs
/Hd981FdVghYYvq//gTakJk0WxfGq030wtXRndPOA0T+qhP3TE+LtGRJ+wARAQAB
wsF1BBgBCgAPBQJXdYAFaHsMBQkHhh+AAAoJEKwQezhmktrdTyEP/0H0VWHwQsaW
jMrGj000MFzXGUo8SBmYYTBs29VM8wBGDsPkYCjeZzU16i9iqDpDqxyqmTigcjH
V8CDx/6xsMBLG2yKaKZ4m3+Yn0Qf/sQkyCvqiyMF9mS7pDYWy+mPhPuw8TDIfiqg
VhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+NAM6Q5dYkCebyvwzLmg1sVni
l6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNViJ9zAaPI78X9v6PpDGn0kg6oL
zrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJB1kke6kw9+KagY8mrVX1ZenRg
+sY/4vxJreYWQeq167ggx+wFjKDcfhZA7m70LH0DysrGVCLcmuinUBaN1HmLDcGY
XZ+kMCoXf0bpuCVByQmNJgEb47EIFlx/+TEeNHKM0+22xL1atFzXfkEVZck+NghL
ZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7GNpuiEFUYh69QQ2//CS5H51o
sC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02NK0fvF/IKHnGkvH28rv00PCv0
WTA/MClv28y0PrSvvcMXnduLtkBEX7TISMPW+n+0Ta63/z4YFFEZ7sFLrEm3Q3vJ
MN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=Z9u3
-----END PGP PUBLIC KEY BLOCK-----
```

# ドキュメント履歴

このトピックでは、AWS SDK for Java デベロッパーガイドの履歴における重要な変更点について説明します。

このガイドは、2024 年 5 月 21 日に最終公開されました。

変更	説明	日付
<a href="#">JVM TTL を設定する方法</a>	Java コマンドラインシステムプロパティを使用してnetworkaddress.cache.ttl セキュリティプロパティを設定する手順を削除します。	2024 年 5 月 21 日
<a href="#">the section called “<u>の SDK の起動時間を短縮する AWS Lambda</u>”</a>	HTTP クライアントのレコメンデーションを更新しての起動時間を短縮する AWS Lambda	2024 年 5 月 14 日
<a href="#">the section called “<u>サービスクライアントのメトリクス</u>”</a>	メトリクステーブル項目を再編成する	2024 年 5 月 1 日
<a href="#">the section called “<u>トラブルシューティング</u>”</a>	トラブルシューティングトピックを追加します。	2024 年 4 月 26 日
<a href="#">the section called “<u>各リクエストで収集されたメトリクス</u>”</a>	SDK によってレポートされる新しいメトリクスを追加します。	2024 年 4 月 26 日
<a href="#">the section called “<u>DNS 名参照用の JVM TTL を設定する</u>”</a>	推奨 DNS ルックアップ TTL を 5 秒に変更します。	2024 年 4 月 23 日
<a href="#">the section called “<u>パッケージ名から artifactId へのマッピング</u>”</a>	Maven artifactId マッピングトピックにパッケージ名を追加します。	2024 年 4 月 17 日

変更	説明	日付
<a href="#">the section called “SDK メトリクスを使用する”</a>	メトリクスセクションに設定の詳細を追加します。	2024 年 4 月 12 日
<a href="#">the section called “IAM Policy Builder API”</a>	IAM Policy Builder API の移行情報を追加します。	2024 年 4 月 11 日
<a href="#">???</a>	HTTP プロキシ情報を更新します。	2024 年 4 月 3 日
<a href="#">the section called “安全な”</a>	IMDSv1 を無効にする手順を追加します。	2024 年 3 月 14 日
<a href="#">the section called “S tep-by-step 命令”</a>	step-by-step 移行手順を追加します。	2024 年 3 月 8 日
<a href="#">バージョン 2 に移行する</a>	移行トピックを更新します。	2024 年 2 月 14 日
<a href="#">the section called “AWS CRT ベースの HTTP クライアントを設定する”</a>	同期 AWS CRT ベースの HTTP クライアントに関する情報を追加します。	2024 年 1 月 5 日
<a href="#">the section called “Amazon Cognito ID”</a> および <a href="#">the section called “Amazon Cognito Identity Provider”</a>	Amazon Cognito の例が「コードの例」セクションに移動しました。	2023 年 12 月 28 日
<a href="#">SDK 機能を使用する</a>	SDK の機能のトピックが変更されました。	2023 年 12 月 11 日
<a href="#">OpenPGP 鍵</a>	現在の OpenPGP 鍵を指定します。	2023 年 12 月 6 日
<a href="#">the section called “シリアル化の変更”</a>	SDK for Java の v1 と v2 のシリアル化の違いを説明します。	2023 年 12 月 5 日

変更	説明	日付
<a href="#">the section called “S3 Transfer Manager”</a>	S3 Transfer Manager のバージョン 1 からバージョン 2 への変更を詳しく説明するセクションを追加。	2023 年 11 月 13 日
<a href="#">the section called “注釈のリアレンス”</a>	DynamoDB 拡張クライアントで使用できるデータクラス注釈のリストを追加。	2023 年 10 月 30 日
<a href="#">???</a>	SDK for Java v1.x から v2.x へのライブラリとユーティリティの統合ステータスに関する情報を追加	2023 年 10 月 17 日
<a href="#">???</a>	Gradle セットアップトピックを更新	2023 年 10 月 17 日
<a href="#">the section called “ネストされたオブジェクトの NULL 属性を無視する”</a>	DynamoDB 拡張クライアント @DynamoDbIgnoreNulls 注釈に関する情報を追加。	2023 年 9 月 22 日
<a href="#">the section called “クロスリージョンアクセス”</a>	Amazon S3 バケットへのクロスリージョンアクセスに関する情報を追加。	2023 年 8 月 31 日
<a href="#">the section called “空のオブジェクトを保持する”</a>	@DynamoDbPreserveEmptyObject 注釈について説明するセクションを追加。	2023 年 8 月 25 日
<a href="#">???</a>	サービスクライアントセクションを更新。	2023 年 8 月 15 日

変更	説明	日付
<a href="#">the section called “クライアントの推奨事項”</a>	バージョン 0.23 以降、AWS CRT は Alpine Linux などの musl ベースの OS をサポートしています。HTTP クライアントの推奨事項に musl サポートを反映。	2023 年 8 月 11 日
<a href="#">the section called “IAM ポリシーを作成する”</a>	IAM Policy Builder API セクションを追加	2023 年 7 月 31 日
<a href="#">the section called “使用を開始する”</a>	「DynamoDB 拡張クライアント」トピックの「はじめに」セクションのいくつかのスニペットを修正。	2023 年 7 月 24 日
<a href="#">the section called “HTTP プロキシの設定”</a>	各 HTTP クライアントの HTTP プロキシサポート情報と例を追加。	2023 年 6 月 2 日
目次の再編成	<a href="#">コードの例</a> セクションおよび <a href="#">の操作 AWS のサービス</a> を最上位の目次項目に掲載。	2023 年 5 月 24 日
<a href="#">the section called “ログ記録を追加する”</a>	Gradle の依存関係をログ記録セクションに表示。	2023 年 5 月 23 日
<a href="#">the section called “ページ分割された結果を処理する”</a>	ページ分割のトピックを更新。	2023 年 5 月 18 日
<a href="#">the section called “Gradle プロジェクトを設定する”</a>	Gradle プロジェクトの設定を更新。	2023 年 5 月 3 日
<a href="#">DynamoDB Enhanced Client API</a>	書き換え DynamoDB Enhanced Client API トピックのリリース。	2023 年 4 月 28 日

変更	説明	日付
<a href="#">更新: 入門ガイドのチュートリアル</a> の説明	Maven アーキタイプが CredentialsProvider のオプションを含むように変更、それに応じインストラクションも変更。	2023 年 4 月 11 日
<a href="#">the section called “クライアントの推奨事項”</a>	HTTP クライアント意思決定ガイダンスを追加	2023 年 3 月 30 日
IAM ベストプラクティスの更新	IAM ベストプラクティスに沿ってガイドを更新しました。詳細については、「 <a href="#">IAM のセキュリティのベストプラクティス</a> 」を参照してください。	2023 年 3 月 14 日
<a href="#">the section called “プロファイルの認証情報を再ロード”</a>	プロファイル認証情報の再ロードに関するセクションを追加。	2023 年 2 月 9 日
<a href="#">the section called “AWS CRT ベースの HTTP クライアントを設定する”</a>	GA リリースのトピックを更新。	2023 年 2 月 8 日
<a href="#">the section called “Amazon EC2 インスタンスメタデータを操作する”</a>	Amazon S3 インスタンスメタデータサービス用の Java SDK クライアントのガイド付き例を追加。	2023 年 2 月 1 日
<a href="#">the section called “パフォーマンスの高い S3 クライアントを使用する”</a>	AWS CRT ベースの S3 クライアントのセクションを追加します。	2022 年 12 月 19 日
<a href="#">the section called “ファイルとディレクトリを転送する”</a>	GA リリース用に Amazon S3 Transfer Manager のサンプルを更新。	2022 年 12 月 19 日

変更	説明	日付
<a href="#">the section called “ベストプラクティス”</a>	ベストプラクティスのセクションを追加。	2022 年 11 月 18 日
<a href="#">the section called “外部プロセスからの一時的な認証情報をロードする”</a>	外部プロセスからの認証情報のロードに関するセクションを追加。	2022 年 11 月 15 日
<a href="#">the section called “サービスクライアントのメトリクス”</a>	HTTP クライアントの使用要件を含むメトリクスリストを更新。	2022 年 11 月 9 日
<a href="#">the section called “ファイルとディレクトリを転送する”</a>	サンプルコードを修正。	2022 年 11 月 2 日
<a href="#">the section called “の SDK の起動時間を短縮する AWS Lambda”</a>	Lambda の起動時間を短縮するためのオプションを追加しセクションを更新。	2022 年 11 月 1 日
<a href="#">the section called “HTTP クライアント”</a>	SDK のすべての HTTP クライアントを対象とする設定情報を追加。	2022 年 10 月 26 日
<a href="#">the section called “ログ記録”</a>	すべての HTTP クライアントのワイヤログ記録の詳細を含むようにログ記録トピックを更新。	2022 年 10 月 4 日
<a href="#">the section called “AWS データベースサービス”</a>	AWS データベースサービスと SDK for Java 2.x の概要セクションを追加しました。	2022 年 9 月 13 日
<a href="#">EC2-Classic Networking は販売終了になります</a>	2022 年 8 月 15 日に、EC2-Classic の提供を終了。	2022 年 7 月 28 日
<a href="#">the section called “追加認証オプション”</a>	シングルサインオン認証に必要な依存関係の更新	2022 年 7 月 18 日



変更	説明	日付
<a href="#">the section called “Transport Layer Security (TLS)”</a>	TLS セキュリティ情報を更新。	2022 年 4 月 8 日
<a href="#">the section called “追加認証オプション”</a>	認証情報の設定および使用に関する詳細を追加。	2021 年 2 月 22 日
<a href="#">the section called “GraalVM Native Image プロジェクトを設定する”</a>	GraalVM Native Image プロジェクトの設定に関する新しいトピック。	2021 年 2 月 18 日
<a href="#">the section called “リソースの状態をポーリングする”</a>	Waiters のリリース: 新しい機能に関するトピックを追加。	2020 年 9 月 30 日
<a href="#">the section called “SDK メトリクスを使用する”</a>	Metrics のリリース: 新しい機能に関するトピックを追加。	2020 年 8 月 17 日
<a href="#">the section called “Amazon SNS”</a>	のサンプルトピックを追加しました Amazon SNS。	2020 年 5 月 30 日
<a href="#">the section called “の SDK の起動時間を短縮する AWS Lambda”</a>	AWS Lambda 関数のパフォーマンスに関するトピックを追加しました。	2020 年 5 月 29 日
<a href="#">the section called “DNS 名参照用の JVM TTL を設定する”</a>	JVM TTL DNS キャッシュに関するトピックを追加。	2020 年 4 月 27 日
<a href="#">the section called “Apache Maven プロジェクトを設定する”, the section called “Gradle プロジェクトを設定する”</a>	新しい Maven と Gradle 設定に関するトピック。	2020 年 4 月 21 日
<a href="#">the section called “Transport Layer Security (TLS)”</a>	セキュリティセクションに TLS 1.2 を追加しました。	2020 年 3 月 19 日
<a href="#">the section called “Amazon Kinesis Data Streams へのサブスクライブ”</a>	Kinesis ストリームの例を追加しました。	2018 年 8 月 2 日

変更	説明	日付
<a href="#">the section called “ページ分割された結果を処理する”</a>	自動ページ分割のトピックを追加しました。	2018 年 4 月 5 日
<a href="#">???</a>	IAM、Amazon EC2、CloudWatch および のサンプルトピックを追加しました DynamoDB。	2017 年 12 月 29 日
<a href="#">the section called “Amazon S3”</a>	の getObject の例を追加しました Amazon S3。	2017 年 8 月 7 日
<a href="#">the section called “非同期プログラミングを使用する”</a>	非同期のトピックを追加しました。	2017 年 8 月 4 日
<a href="#">AWS SDK for Java 2.x の一般提供リリース</a>	AWS SDK for Java バージョン 2 (v2) がリリースされました。	2017 年 2 月 6 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。