デベロッパーガイド

AWS SDK for Kotlin

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS SDK for Kotlin: デベロッパーガイド

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスはAmazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

とは AWS SDK for Kotlin 1
SDK の使用を開始する 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 1
はじめに 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 6
ステップ 4: アプリケーションをビルドして実行する 8
Success (成功) 9
クリーンアップ 9
次のステップ 9
セットアップする 10
基本的なセットアップ 10
概要 10
AWS アクセスポータルへのサインイン機能 11
シングルサインオンを設定する 12
を使用してサインインする AWS CLI 13
Java と構築ツールをインストールする 13
一時認証情報の使用 14
プロジェクトビルドファイルを作成する 15
プロジェクトのコーディング 20
を使用してログインする AWS CLI 21
構成する 22
サービスクライアントを作成する 24
コードでクライアントを設定する 24
環境からクライアントを設定する 25
クライアントを閉じる 26
AWS リージョン 選択 27
デフォルトのリージョンプロバイダーチェーン 27
認証情報プロバイダー 27
デフォルトの認証情報プロバイダーチェーン 28
明示的な認証情報プロバイダー 30
クライアントエンドポイント 31

カスタム設定 32
例 35
HTTP 36
HTTP クライアント設定 36
HTTP プロキシを使用する 40
インターセプター 40
最小 TLS バージョンを強制する 42
再試行 43
デフォルト設定 44
最大試行回数 44
遅延とバックオフ 45
トークンバケットを再試行する 47
アダプティブ再試行 50
オブザーバビリティ 51
を設定する TelemetryProvider 52
メトリクス 53
ログ記録 56
テレメトリプロバイダー 59
クライアント設定を上書きする 61
オーバーライドされたクライアントライフサイクル 62
共有 リソース 62
SDK を使用する 64
リクエストを発行する 64
サービスインターフェイスの DSL オーバーロード 65
入力を必要としないリクエスト 66
コルーチン 66
同時リクエストの実行 66
ブロックリクエストの作成 67
ストリーミング操作 68
ストリーミングレスポンス 68
ストリーミングリクエスト 69
ページ分割 70
ウェーター 71
エラー処理 72
サービス例外 72
クライアントの例外 72

エラーメタデータ ... 73
リクエストの事前署名 ... 73
署名の基本 ... 74
高度な事前署名設定 ... 74
POST リクエストと PUT リクエストの事前署名 ... 75
SDK が事前署名できるオペレーション ... 76
トラブルシューティングに関するよくある質問 ... 76
「接続が閉じられた」問題を修正するにはどうすればよいですか？ ... 77
最大試行回数に達する前に例外がスローされるのはなぜですか？ ... 77
NoSuchMethodError または NoClassDefFoundError を修正するにはどうすればよいですか？ ... 78
依存関係の競合を解決するにはどうすればよいですか？ ... 79
モッキング ... 81
MockK ... 81
の使用 AWS のサービス ... 87
Amazon S3 ... 88
チェックサムによるデータ整合性保護 ... 89
マルチリージョンアクセスポイントの使用 ... 93
DynamoDB ... 99
AWS アカウントベースのエンドポイントを使用する ... 99
DynamoDB マッパーを使用する (開発者プレビュー) ... 100
コードの例 ... 126
API Gateway ... 127
シナリオ ... 128
Aurora ... 128
基本 ... 129
アクション ... 142
シナリオ ... 128
Auto Scaling ... 156
基本 ... 129
アクション ... 142
Amazon Bedrock ... 174
アクション ... 142
Amazon Bedrock ランタイム ... 175
Amazon Nova ... 175
Amazon Titan Text ... 180

CloudWatch 182
基本 129
アクション 142
CloudWatch Logs 223
アクション 142
Amazon Cognito Identity Provider 226
アクション 142
シナリオ 128
Amazon Comprehend 242
シナリオ 128
DynamoDB 243
基本 129
アクション 142
シナリオ 128
Amazon EC2 272
基本 129
アクション 142
Amazon ECR 303
基本 129
アクション 142
OpenSearch Service 333
アクション 142
EventBridge 337
基本 129
アクション 142
AWS Glue 368
基本 129
アクション 142
IAM 379
基本 129
アクション 142
AWS IoT 399
基本 129
アクション 142
AWS IoT data 423
アクション 142

AWS IoT FleetWise ... 425
基本 ... 129
アクション ... 142
Amazon Keyspaces ... 460
基本 ... 129
アクション ... 142
AWS KMS ... 486
アクション ... 142
Lambda ... 495
基本 ... 129
アクション ... 142
シナリオ ... 128
Amazon Location ... 505
基本 ... 129
アクション ... 142
MediaConvert ... 536
アクション ... 142
Amazon Pinpoint ... 550
アクション ... 142
Amazon RDS ... 560
基本 ... 129
アクション ... 142
シナリオ ... 128
Amazon RDS データサービス ... 578
シナリオ ... 128
Amazon Redshift ... 579
アクション ... 142
シナリオ ... 128
Amazon Rekognition ... 583
アクション ... 142
シナリオ ... 128
Route 53 ドメイン登録 ... 602
基本 ... 129
アクション ... 142
Amazon S3 ... 621
基本 ... 129

アクション 142
シナリオ 128
SageMaker AI 643
アクション 142
シナリオ 128
Secrets Manager 668
アクション 142
Amazon SES 669
シナリオ 128
Amazon SNS 672
アクション 142
シナリオ 128
Amazon SQS 699
アクション 142
シナリオ 128
Step Functions 721
基本 129
アクション 142
サポート 743
基本 129
アクション 142
Amazon Translate 761
シナリオ 128
セキュリティ 763
データ保護 763
TLS 1.2 の適用 764
Java での TLS のサポート 764
TLS のバージョンを確認する方法 765
Identity and Access Management 765
対象者 765
アイデンティティを使用した認証 766
ポリシーを使用したアクセスの管理 770
IAM AWS のサービス の操作方法 772
AWS ID とアクセスのトラブルシューティング 773
コンプライアンス検証 775
耐障害性 776

インフラストラクチャセキュリティ 776
ドキュメント履歴 778
..... dcclxxxii

とは AWS SDK for Kotlin

AWS SDK for Kotlin は、Amazon Web Services 用の Kotlin APIs。SDK を使用すると、Amazon S3、Amazon EC2Amazon EC2 DynamoDB などで動作する Kotlin アプリケーションを構築できます。Kotlin SDK を使用すると、JVM プラットフォームまたは Android API レベル 24 以上をターゲットにできます。JavaScript や Native などの追加のプラットフォームのサポートは、今後のリリースで予定されています。

今後のリリースで予定されている機能を追跡するには、GitHub のロードマップを参照してください。

SDK の使用を開始する

SDK の使用を開始するには、 はじめにチュートリアルに従います。

開発環境を設定するには、「セットアップする」を参照してください。

へのリクエストを行うためのサービスクライアントを作成および設定するには AWS のサービス、「設定」を参照してください。SDK のさまざまな機能については、「」を参照してください SDK を使用する。

特定の API オペレーションを実行するユースケースと例については、「」を参照してくださいコードの例。

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 SDKsメンテナンスポリシー
- AWS SDKsとツールのバージョンサポートマトリックス

追加リソース

このガイドに加えて、SDK for Kotlin 開発者にとって重要なオンラインリソースを以下に示します。

- AWS 開発者ブログ

- 開発者フォーラム
- SDK ソース (GitHub)
- AWS Code Sample Catalog
- @awsdevelopers (X、以前の Twitter)

SDK for Kotlin の使用を開始する

AWS SDK for Kotlin には、それぞれの Kotlin APIsが用意されています AWS のサービス。SDK を使用すると、Amazon S3、Amazon EC2Amazon EC2、Amazon DynamoDB などで動作する Kotlin 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、Gradle を使用して の依存関係を定義する方法を示します AWS SDK for Kotlin。次に、DynamoDB テーブルにデータを書き込むコードを作成します。IDE の機能を使用することもできますが、このチュートリアルに必要なのはターミナルウィンドウとテキストエディタだけです。

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、DynamoDB にアクセスできる IAM Identity Center アクセス許可セットと、IAM Identity Center のシングルサインオン設定で構成された Kotlin 開発環境が必要です AWS。

このチュートリアルの基本設定を取得するには、このガイド基本的なセットアップの「」の手順に従います。

Kotlin 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取得したら、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のプロジェクトを作成するには、まず Gradle を使用して Kotlin プロジェクトの基本的なファイルを作成します。次に、 に必要な設定、依存関係、コードでファイルを更新します AWS SDK for Kotlin。

Gradle を使用して新しいプロジェクトを作成するには

> **Note**
>
> このチュートリアルでは、以下のステップ 3 で 5 つのプロンプトを提供する `gradle init` コマンドで Gradle バージョン 8.11.1 を使用します。別のバージョンの Gradle を使用すると、プロンプトと、事前入力されたアーティファクトのバージョンが異なる場合があります。

1. デスクトップやホームフォルダなど、任意の`getstarted`場所に という名前の新しいディレクトリを作成します。
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き、作成した`getstarted`ディレクトリに移動します。
3. 次のコマンドを使用して、新しい Gradle プロジェクトと基本的な Kotlin クラスを作成します。

```
gradle init --type kotlin-application --dsl kotlin
```

   - ターゲット の入力を求められたら`Java version`、 を押します `Enter` (デフォルトは 21)。
   - を求められたら`Project name`、 を押します `Enter` (このチュートリアル`getstarted`では、デフォルトでディレクトリ名になります)。
   - の入力を求められたら`application structure`、 を押します `Enter` (デフォルトは `Single application project`)。
   - とプロンプトが表示されたら`Select test framework`、 を押します `Enter` (デフォルトは `kotlin.test`)。
   - とプロンプトが表示されたら`Generate build using new APIs and behavior`、 を押します `Enter` (デフォルトは no)。

と Amazon S3 の AWS SDK for Kotlin 依存関係を使用してプロジェクトを設定するには

- 前の手順で作成した`getstarted`ディレクトリで、`settings.gradle.kts`ファイルの内容を次のコンテンツに置き換え、*`X.Y.Z`* を最新バージョンの SDK for Kotlin に置き換えます。

```
dependencyResolutionManagement {
    repositories {
        mavenCentral()
    }
```

```
    versionCatalogs {
        create("awssdk") {
            from("aws.sdk.kotlin:version-catalog:X.Y.Z")
        }
    }
}

plugins {
    // Apply the foojay-resolver plugin to allow automatic download of JDKs.
    id("org.gradle.toolchains.foojay-resolver-convention") version "0.8.0"
}

rootProject.name = "getstarted"
include("app")
```

- gradle ディレクトリ内のgetstartedディレクトリに移動します。という名前のバージョンカタログファイルの内容を次の内容libs.versions.tomlに置き換えます。

```
[versions]
junit-jupiter-engine = "5.10.3"

[libraries]
junit-jupiter-engine = { module = "org.junit.jupiter:junit-jupiter-engine",
 version.ref = "junit-jupiter-engine" }

[plugins]
kotlin-jvm = { id = "org.jetbrains.kotlin.jvm", version = "2.1.0" }
```

- app ディレクトリに移動し、build.gradle.kts ファイルを開きます。その内容を次のコードに置き換えて、変更を保存します。

```
plugins {
    alias(libs.plugins.kotlin.jvm)
    application
}


dependencies {
    implementation(awssdk.services.s3) // Add dependency on the AWS SDK for Kotlin's
 S3 client.

    testImplementation("org.jetbrains.kotlin:kotlin-test-junit5")
    testImplementation(libs.junit.jupiter.engine)
```

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

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(21)
    }
}

application {
    mainClass = "org.example.AppKt"
}

tasks.named<Test>("test") {
    useJUnitPlatform()
}
```

dependencies セクションには、 の Amazon S3 モジュールのimplementationエントリが含まれています AWS SDK for Kotlin。Gradle コンパイラは、 javaセクションで Java 21 を使用するように設定されています。

ステップ 3: コードを記述する

プロジェクトを作成して設定したら、次のサンプルコードを使用するAppようにプロジェクトのデフォルトクラスを編集します。

1. プロジェクトフォルダ app で、ディレクトリ src/main/kotlin/org/example に移動します。App.kt ファイルを開きます。
2. その内容を次のコードに置き換え、ファイルを保存します。

```
package org.example

import aws.sdk.kotlin.services.s3.*
import aws.sdk.kotlin.services.s3.model.BucketLocationConstraint
import aws.smithy.kotlin.runtime.content.ByteStream
import kotlinx.coroutines.runBlocking
import java.util.UUID

val REGION = "us-west-2"
val BUCKET = "bucket-${UUID.randomUUID()}"
val KEY = "key"
```

```
fun main(): Unit = runBlocking {
    S3Client
        .fromEnvironment { region = REGION }
        .use { s3 ->
            setupTutorial(s3)

            println("Creating object $BUCKET/$KEY...")

            s3.putObject {
                bucket = BUCKET
                key = KEY
                body = ByteStream.fromString("Testing with the Kotlin SDK")
            }

            println("Object $BUCKET/$KEY created successfully!")

            cleanUp(s3)
        }
}

suspend fun setupTutorial(s3: S3Client) {
    println("Creating bucket $BUCKET...")
    s3.createBucket {
        bucket = BUCKET
        if (REGION != "us-east-1") { // Do not set location constraint for us-east-1.
            createBucketConfiguration {
                locationConstraint = BucketLocationConstraint.fromValue(REGION)
            }
        }
    }
    println("Bucket $BUCKET created successfully!")
}

suspend fun cleanUp(s3: S3Client) {
    println("Deleting object $BUCKET/$KEY...")
    s3.deleteObject {
        bucket = BUCKET
        key = KEY
    }
    println("Object $BUCKET/$KEY deleted successfully!")

    println("Deleting bucket $BUCKET...")
    s3.deleteBucket {
```

```
        bucket = BUCKET
    }
    println("Bucket $BUCKET deleted successfully!")
}
```

ステップ 4: アプリケーションをビルドして実行する

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

1. 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。
2. 次のコマンドを使用して、アプリケーションを構築して実行します。

```
gradle run
```

> **Note**
>
> を取得した場合`IdentityProviderException`、アクティブなシングルサインオンセッションがない可能性があります。`aws sso login` AWS CLI コマンドを実行して、新しいセッションを開始します。

アプリケーションは createBucket API オペレーションを呼び出して新しい S3 バケットを作成し、putObject を呼び出して新しいオブジェクトを新しい S3 バケットに配置します。

末尾の `cleanUp()`関数では、アプリケーションはオブジェクトを削除し、S3 バケットを削除します。

Amazon S3 コンソールで結果を表示するには

1. `cleanUp(s3)` で`App.kt`、 `runBlocking`セクションの行をコメントアウトし、ファイルを保存します。
2. プロジェクトを再構築し、 を実行して新しいオブジェクトを新しい S3 バケットに配置しますgradle run。
3. Amazon S3 コンソールにサインインして、新しい S3 バケットの新しいオブジェクトを表示します。

オブジェクトを表示したら、S3 バケットを削除します。

Success (成功)

Gradle 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、おめでとうございます。を使用して最初の Kotlin アプリケーションを正常に構築しました AWS SDK for Kotlin。

クリーンアップ

新しいアプリケーションの開発が完了したら、このチュートリアルで作成した AWS リソースをすべて削除して、料金が発生しないようにします。ステップ 2 で作成したプロジェクトフォルダ (`get-started`) を削除またはアーカイブすることもできます。

リソースをクリーンアップするには、次の手順に従います。

- `cleanUp()` 関数への呼び出しをコメントアウトした場合は、Amazon S3 コンソールを使用して S3 バケットを削除します。 Amazon S3

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。

- SDK for Kotlin を使用するための追加のセットアップ手順
- SDK for Kotlin の設定
- SDK for Kotlin の使用
- SDK for Kotlin のセキュリティ

のセットアップ AWS SDK for Kotlin

AWS のサービス を使用して にリクエストを行うには AWS SDK for Kotlin、以下が必要です。

- AWS アクセスポータルにサインインする機能
- アプリケーションに必要な AWS リソースを使用するアクセス許可
- 次の要素を備えた開発環境:
  - 次のいずれかの方法で設定されている共有設定ファイル。
    - `config` ファイルには、SDK が認証情報を取得できるように IAM Identity Center の AWS 認証情報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
  - Gradle や Maven などのビルド自動化ツール
- 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

このトピックの内容

- 基本的なセットアップ
- プロジェクトビルドファイルを作成する
- SDK for Kotlin を使用して Kotlin プロジェクトをコーディングする

基本的なセットアップ

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 Kotlin、次の要件を満たす必要があります。

- AWS IAM Identity Centerにある AWS アクセスポータルにサインイン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、 AWS のサービス アプリケーション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するアクセス許可は、ほとんどの開発ニーズに十分対応できます。
- 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するための 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、ただし推奨) IntelliJ IDEA や Eclipse などの IDE (統合開発環境）。

  IDE を使用すると、 AWS Toolkitを統合して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、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ら、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 Identity Center とは独立して動作する外部 ID プロバイダーを使用している場合は、 SDK for Kotlin が使用する一時的な認証情報を取得する方法について説明します。以下の手順に従って、一時的な認証情報を `~/.aws/credentials` ファイルに追加します。
> 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。

AWS アクセスポータルへのサインイン機能

AWS アクセスポータルは、IAM アイデンティティセンターに手動でサインインするウェブロケーションです。URL の形式は `d-xxxxxxxxxx.awsapps.com/start`または `yourdomain.awsapps.com/start`。

AWS アクセスポータルに慣れていない場合は、「SDK およびツールリファレンスガイド」の「IAM Identity Center 認証」トピックのアカウントアクセスに関するガイダンスに従ってください。 AWS SDKs

SDK のシングルサインオンアクセスを設定する

SDK が IAM Identity Center 認証を使用す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 を完了したら、システムに次の要素が含まれているはずです。

- 。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するために使用します。
- デフォルトプロファイルを含む ~/.aws/config ファイル。SDK for Kotlin は、 にリクエストを送信する前に、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 AWS。IAM Identity Center 許可セットに接続された IAM ロールである sso_role_name 値により、アプリケーションで使用されている AWS のサービス にアクセスできます。

  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には、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設定があるため、 `--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が使用されている場合、コマンドは ですaws sso login --profile *named-profile*。

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って aws sso login を実行している場合は、認証情報を入力するように要求されません。
> 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。
> および AWS CLI SDK for Kotlin の情報へのアクセスを許可する を選択します。

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。

- JDK 8 以降。は、Oracle Java SE Development Kit と、、Red Hat OpenJDKAmazon Corretto、AdoptOpenJDK などの Open Java Development Kit (OpenJDK) のディストリビューションで AWS SDK for Kotlin 動作します。 OpenJDK
- 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する方法については、http://maven.apache.org/ を参照してください。
- Gradle をインストールして使用する方法については、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する方法については、https://www.jetbrains.com/idea/ を参照してください。

一時認証情報の使用

SDK の IAM Identity Center シングルサインオンアクセスを設定する代わりに、一時的な認証情報を使用して開発環境を設定できます。

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、作業中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>
```

3. ファイルを保存します。これで、ファイル ~/.aws/credentials はローカルの開発システムに存在しているはずです。このファイルには、特定の名前付きプロファイルが指定されていない場合に SDK for Kotlin が使用する [デフォルト〕 プロファイルが含まれています。
4. AWS アクセスポータルにサインインする
5. アクセス AWS ポータルから IAM ロールの認証情報をコピーするには、「認証情報の手動更新」の見出しにある手順に従ってください。

   a. リンク先の手順のステップ 4 で、開発ニーズに合ったアクセスを許可する IAM ロールの名前を選択します。通常、このロールには PowerUserAccess や Developer などの名前が付いています。

   b. ステップ 7 で、[AWS 認証情報ファイルにプロファイルを手動追加] オプションを選択し、内容をコピーします。

6. コピーした認証情報をローカル credentials ファイルに貼り付け、生成されたプロファイル名を削除します。ファイルは以下のようになります。

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

7. `credentials` ファイルを保存します。

SDK for Kotlin は、サービスクライアントを作成し、各リクエストに使用するときに、これらの一時的な認証情報にアクセスします。ステップ 5a で選択した IAM ロールの設定により、一時的な認証情報の有効期間が決まります。最大期間は 12 時間です。

一時的な認証情報の有効期限が切れたら、ステップ 4～7 を繰り返します。

プロジェクトビルドファイルを作成する

シングルサインオンアクセスと開発環境を設定したら、任意のビルドツールを使用して Kotlin プロジェクトを作成します。ビルドファイルで、アプリケーション AWS のサービス がアクセスする必要がある の依存関係を指定します。

考えられるすべての Maven アーティファクト名のリストを確認するには、 API リファレンスドキュメントを参照してください。SDK の最新バージョンを確認するには、GitHub で最新リリースを確認してください。

次のサンプルビルドファイルは、7 つの を使用してプロジェクトのコーディングを開始するために必要な要素を提供します AWS のサービス。

Gradle

は、依存関係の名前を検出し、複数のアーティファクト間でバージョン番号を同期させるのに役立つ Gradle バージョンカタログと部品表 (BOM) AWS SDK for Kotlin を発行します。

バージョンカタログは、バージョン 8 より前の Gradle のプレビュー機能であることに注意してください。使用する Gradle のバージョンによっては、特徴量プレビュー API を使用してオプトインする必要がある場合があります。

Gradle バージョンカタログを使用するには

1. `settings.gradle.kts` ファイルで、 `versionCatalogs`ブロック内に `dependencyResolutionManagement`ブロックを追加します。

次のサンプルファイルは、 AWS SDK for Kotlin バージョンカタログを設定します。*X.Y.Z* リンクに移動して、利用可能な最新バージョンを確認できます。

```
plugins {
    id("org.gradle.toolchains.foojay-resolver-convention") version "X.Y.Z"
}
rootProject.name = "your-project-name"

dependencyResolutionManagement {
    repositories {
        mavenCentral()
    }

    versionCatalogs {
        create("awssdk") {
            from("aws.sdk.kotlin:version-catalog:X.Y.Z")
        }
    }
}
```

2. バージョンカタログで利用可能なタイプセーフ識別子`build.gradle.kts`を使用して、 の依存関係を宣言します。

   次のサンプルファイルは、7 つの の依存関係を宣言します AWS のサービス。

```
plugins {
    kotlin("jvm") version "X.Y.Z"
    application
}

group = "org.example"
version = "1.0-SNAPSHOT"

repositories {
    mavenCentral()
}

dependencies {
    implementation(platform(awssdk.bom))
    implementation(platform("org.apache.logging.log4j:log4j-bom:X.Y.Z"))

    implementation(awssdk.services.s3)
```

```
    implementation(awssdk.services.dynamodb)
    implementation(awssdk.services.iam)
    implementation(awssdk.services.cloudwatch)
    implementation(awssdk.services.cognitoidentityprovider)
    implementation(awssdk.services.sns)
    implementation(awssdk.services.pinpoint)
    implementation("org.apache.logging.log4j:log4j-slf4j2-impl")

    // Test dependency.
    testImplementation(kotlin("test"))
}

tasks.test {
    useJUnitPlatform()
}

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(X*)
    }
}

application {
    mainClass = "org.example.AppKt"
}
```

*Java バージョン。例: 17または 21。

Maven

次のサンプルpom.xmlファイルには、7 つの の依存関係があります AWS のサービス。*X.Y.Z* リンクに移動して、利用可能な最新バージョンを確認できます。

```
<?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/maven-v4_0_0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
```

```
    <artifactId>setup</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <aws.sdk.kotlin.version>X.Y.Z</aws.sdk.kotlin.version>
        <kotlin.version>X.Y.Z</kotlin.version>
        <log4j.version>X.Y.Z</log4j.version>
        <junit.jupiter.version>X.Y.Z</junit.jupiter.version>
        <jvm.version>X*</jvm.version>
    </properties>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>aws.sdk.kotlin</groupId>
                <artifactId>bom</artifactId>
                <version>${aws.sdk.kotlin.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <dependency>
                <groupId>org.apache.logging.log4j</groupId>
                <artifactId>log4j-bom</artifactId>
                <version>${log4j.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>

        </dependencies>
    </dependencyManagement>

    <dependencies>
        <dependency>
            <groupId>aws.sdk.kotlin</groupId>
            <artifactId>s3-jvm</artifactId>
        </dependency>
        <dependency>
            <groupId>aws.sdk.kotlin</groupId>
            <artifactId>dynamodb-jvm</artifactId>
        </dependency>
        <dependency>
            <groupId>aws.sdk.kotlin</groupId>
            <artifactId>iam-jvm</artifactId>
        </dependency>
```

```
        <dependency>
            <groupId>aws.sdk.kotlin</groupId>
            <artifactId>cloudwatch-jvm</artifactId>
        </dependency>
        <dependency>
            <groupId>aws.sdk.kotlin</groupId>
            <artifactId>cognitoidentityprovider-jvm</artifactId>
        </dependency>
        <dependency>
            <groupId>aws.sdk.kotlin</groupId>
            <artifactId>sns-jvm</artifactId>
        </dependency>
        <dependency>
            <groupId>aws.sdk.kotlin</groupId>
            <artifactId>pinpoint-jvm</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-slf4j2-impl</artifactId>
        </dependency>

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

    <build>
        <sourceDirectory>src/main/kotlin</sourceDirectory>
        <testSourceDirectory>src/test/kotlin</testSourceDirectory>

        <plugins>
            <plugin>
                <groupId>org.jetbrains.kotlin</groupId>
                <artifactId>kotlin-maven-plugin</artifactId>
```

```
                <version>${kotlin.version}</version>
                <executions>
                    <execution>
                        <id>compile</id>
                        <phase>compile</phase>
                        <goals>
                            <goal>compile</goal>
                        </goals>
                    </execution>
                    <execution>
                        <id>test-compile</id>
                        <phase>test-compile</phase>
                        <goals>
                            <goal>test-compile</goal>
                        </goals>
                    </execution>
                </executions>
                <configuration>
                    <jvmTarget>${jvm.version}</jvmTarget>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

*Java バージョン。例: 17または 21。

SDK for Kotlin を使用して Kotlin プロジェクトをコーディングする

これで、楽しさが始まります。アプリケーションを開発するときは、AWS SDK for Kotlin API オペレーションの詳細については、API リファレンスを参照してください。一般的な Kotlin API 情報には、次のリンクを使用します。

- 標準ライブラリ API リファレンス
- コルーチンの概要
- Coroutines API

を使用してログインする AWS CLI

アクセスするプログラムを実行するたびに AWS のサービス、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。この操作はは、次の コマンドを実行して行うことができます。

```
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。IAM Identity Center シングルサインオン設定で名前付きプロファイルが使用されている場合、コマンドは ですaws sso login --profile named-profile。

アクティブなセッションが既に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。

を設定する AWS SDK for Kotlin

このセクションでは、 を使用してサービスクライアントを設定する方法について説明します AWS SDK for Kotlin。詳細については、 SDK およびツールリファレンスガイドを参照してください。これには、すべての AWS SDKs に適用される設定の概要が含まれています。

目次

- サービスクライアントを作成する
  - コードでクライアントを設定する
  - 環境からクライアントを設定する
  - クライアントを閉じる
- AWS リージョン 選択
  - デフォルトのリージョンプロバイダーチェーン
- 認証情報プロバイダー
  - デフォルトの認証情報プロバイダーチェーン
    - デフォルトの認証情報プロバイダーチェーンについて説明します。
  - 明示的な認証情報プロバイダー
- クライアントエンドポイントを設定する
  - カスタム設定
    - endpointUrl を設定する
    - endpointProvider を設定する
      - EndpointProvider のプロパティ
    - endpointUrl、または endpointProvider
    - Amazon S3 に関する注意事項
  - 例
    - endpointUrl の例
    - endpointProvider の例
    - endpointUrl および endpointProvider
- HTTP
  - HTTP クライアント設定
    - 基本的な設定

        - インポート
        - コード
    - HTTP エンジンタイプの指定
        - インポート
        - コード
        - OkHttp4Engine を使用する
        - 明示的な HTTP クライアントを使用する
            - インポート
            - コード
- HTTP プロキシを使用する
    - JVM システムプロパティを使用する
    - 環境変数を使用します。
    - EC2 インスタンスでプロキシを使用する
- HTTP インターセプター
    - インターセプターの登録
        - すべてのサービスクライアントオペレーションのインターセプター
        - 特定のオペレーションのみのインターセプター
- 最小 TLS バージョンを強制する
    - HTTP エンジンを設定する
    - sdk.minTls JVM システムプロパティを設定する
    - SDK_MIN_TLS 環境変数を設定する
- 再試行
    - デフォルトの再試行設定
    - 最大試行回数
    - 遅延とバックオフ
    - トークンバケットを再試行する
    - アダプティブ再試行
- オブザーバビリティ
- を設定する TelemetryProvider
    - デフォルトのグローバルテレメトリプロバイダーを設定する

        - 特定のサービスクライアントのテレメトリプロバイダーを設定する
    - メトリクス
    - ログ記録
        - ワイヤレベルのメッセージのログモードを指定する
            - コードでログモードを設定する
            - 環境からログモードを設定する
    - テレメトリプロバイダー
        - OpenTelemetry ベースのテレメトリプロバイダーを設定する
            - 前提条件
            - SDK を設定
            - リソース
- サービスクライアント設定を上書きする
    - オーバーライドされたクライアントのライフサイクル
    - クライアント間で共有されるリソース

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、まずそのサービスのクライアントをインスタンス化する必要があります。

使用する HTTP クライアント、ログ記録レベル、再試行設定など、サービスクライアントの共通設定を設定できます。さらに、各サービスクライアントには、 AWS リージョン と認証情報プロバイダーが必要です。SDK はこれらの値を使用して、正しいリージョンにリクエストを送信し、正しい認証情報を使用してリクエストに署名します。

これらの値は、プログラムでコードで指定することも、環境から自動的にロードすることもできます。

コードでクライアントを設定する

特定の値でサービスクライアントを設定するには、次のスニペットに示すように、サービスクライアントファクトリメソッドに渡される Lambda 関数で指定できます。

```
val dynamoDbClient = DynamoDbClient {
    region = "us-east-1"
```

```
    credentialsProvider = ProfileCredentialsProvider(profileName = "myprofile")
}
```

設定ブロックで指定しない値は、デフォルトに設定されます。たとえば、前のコードのように認証情報プロバイダーを指定しない場合、認証情報プロバイダーはデフォルトでデフォルトの認証情報プロバイダーチェーンになります。

> **Warning**
>
> などの一部のプロパティにはデフォルトregionがありません。プログラムによる設定を使用する場合は、設定ブロックで明示的に指定する必要があります。SDK が プロパティを解決できない場合、API リクエストが失敗する可能性があります。

環境からクライアントを設定する

サービスクライアントを作成すると、SDK は現在の実行環境内の場所を検査し、いくつかの設定プロパティを決定できます。これらの場所には、共有設定ファイルと認証情報ファイル、環境変数、JVM システムプロパティが含まれます。解決できるプロパティには、AWS リージョン、再試行戦略、ログモードなどがあります。SDK が実行環境から解決できるすべての設定の詳細については、AWS SDKs」を参照してください。

環境ソース設定でクライアントを作成するには、サービスクライアントインターフェイスsuspend fun fromEnvironment()で静的メソッドを使用します。

```
val dynamoDbClient = DynamoDbClient.fromEnvironment()
```

この方法でクライアントを作成すると、Amazon EC2 で を実行する場合 AWS Lambdaや、サービスクライアントの設定を環境から利用できるその他のコンテキストで実行する場合に便利です。これにより、コードが実行中の環境から切り離され、コードを変更せずにアプリケーションを複数のリージョンにデプロイすることが容易になります。

さらに、Lambda ブロックを に渡すことで、特定のプロパティを上書きできますfromEnvironment。次の例では、環境 (リージョンなど) からいくつかの設定プロパティをロードしますが、特にプロファイルから認証情報を使用するように認証情報プロバイダーを上書きします。

```
val dynamoDbClient = DynamoDbClient.fromEnvironment {
    credentialsProvider = ProfileCredentialsProvider(profileName = "myprofile")
```

```
}
```

SDK は、プログラム設定または環境から決定できない設定プロパティにデフォルト値を使用します。たとえば、コードまたは環境設定で認証情報プロバイダーを指定しない場合、認証情報プロバイダーはデフォルトでデフォルトの認証情報プロバイダーチェーンになります。

> **Warning**
>
> リージョンなどの一部のプロパティにはデフォルトがありません。環境設定で指定するか、設定ブロックで明示的に指定する必要があります。SDK が プロパティを解決できない場合、API リクエストが失敗する可能性があります。

> **Note**
>
> 一時的なアクセスキーや SSO 設定などの認証情報関連のプロパティは実行環境で見つかりますが、値は作成時にクライアントによって取得されません。代わりに、値はリクエストごとに認証情報プロバイダーレイヤーによってアクセスされます。

クライアントを閉じる

サービスクライアントが不要になった場合は、サービスクライアントを閉じて、使用しているリソースを解放します。

```
val dynamoDbClient = DynamoDbClient.fromEnvironment()
// Invoke several DynamoDB operations.
dynamoDbClient.close()
```

サービスクライアントはCloseableインターフェイスを拡張するため、次のスニペットに示すように、 use拡張機能を使用してブロックの最後にクライアントを自動的に閉じることができます。

```
DynamoDbClient.fromEnvironment().use { dynamoDbClient ->
    // Invoke several DynamoDB operations.
}
```

前の例では、Lambda ブロックは、先ほど作成されたクライアントへの参照を受け取ります。このクライアントリファレンスでオペレーションを呼び出すことができ、例外をスローするなどしてブロックが完了すると、クライアントは閉じられます。

AWS リージョン 選択

を使用すると AWS リージョン、特定の地域 AWS のサービス で動作する にアクセスできます。これは、冗長性と、ユーザーがアクセスする場所の近くでのデータとアプリケーションの実行を維持するために有効です。

デフォルトのリージョンプロバイダーチェーン

環境からサービスクライアントの設定をロードする場合、次のルックアッププロセスが使用されます。

1. ビルダーに設定された明示的なリージョン。
2. `aws.region` JVM システムプロパティがチェックされます。設定されている場合、そのリージョンはクライアントの設定で使用されます。
3. `AWS_REGION` 環境変数が確認されます。設定されている場合、そのリージョンはクライアントの設定で使用されます。
   a. 注: この環境変数は Lambda コンテナによって設定されます。
4. SDK は、 AWS 共有設定ファイルをチェックします。`region` プロパティがアクティブなプロファイルに設定されている場合、SDK はそれを使用します。
   a. `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。
   b. `aws.profile` JVM システムプロパティまたは`AWS_PROFILE`環境変数を使用して、SDK がロードするプロファイルをカスタマイズできます。
5. SDK は、Amazon EC2 インスタンスメタデータサービスを使用して、現在実行中の EC2 インスタンスのリージョンを決定しようとします。
6. この時点でリージョンがまだ解決されていない場合、クライアントの作成は例外で失敗します。

認証情報プロバイダー

⚠ デフォルトの認証情報プロバイダーチェーンがバージョン 1.4.0 で変更された認証情報を解決する順序 詳細については、以下の注意事項を参照してください。

を使用して Amazon Web Services にリクエストを送信する場合 AWS SDK for Kotlin、リクエストはによって発行された認証情報で暗号的に署名される必要があります AWS。Kotlin SDK は、自動的にリクエストに署名します。認証情報を取得するために、SDK は JVM システムプロパティ、環境変数、共有 AWS `config` ファイルと `credentials` ファイル、Amazon EC2 インスタンスメタデータなど、いくつかの場所にある設定を使用できます。

SDK は、認証情報プロバイダーの抽象化を使用して、さまざまなソースから認証情報を取得するプロセスを簡素化します。SDK には、いくつかの認証情報プロバイダーの実装が含まれています。

例えば、取得された設定に共有`config`ファイルからの IAM Identity Center シングルサインオンアクセス設定が含まれている場合、SDK は IAM Identity Center と連携して、リクエストに使用する一時的な認証情報を取得します AWS のサービス。認証情報を取得するこのアプローチでは、SDK は IAM Identity Center プロバイダー (SSO 認証情報プロバイダーとも呼ばれます) を使用します。このガイドのセットアップセクションでは、この設定について説明しました。

特定の認証情報プロバイダーを使用するには、サービスクライアントを作成するときに指定できます。または、デフォルトの認証情報プロバイダーチェーンを使用して、設定を自動的に検索することもできます。

デフォルトの認証情報プロバイダーチェーン

クライアントの構築時に明示的に指定しない場合、SDK for Kotlin は認証情報プロバイダーを使用して、認証情報を提供できる各場所を順番にチェックします。このデフォルトの認証情報プロバイダーは、認証情報プロバイダーのチェーンとして実装されます。

デフォルトのチェーンを使用してアプリケーションで認証情報を指定するには、`credentialsProvider`プロパティを明示的に指定せずにサービスクライアントを作成します。

```
val ddb = DynamoDbClient {
    region = "us-east-2"
}
```

サービスクライアントの作成の詳細については、「クライアントの構築と設定」を参照してください。

デフォルトの認証情報プロバイダーチェーンについて説明します。

デフォルトの認証情報プロバイダーチェーンは、次の事前定義されたシーケンスを使用して認証情報設定を検索します。設定された設定が有効な認証情報を提供すると、チェーンは停止します。

1. AWS アクセスキー (JVM システムプロパティ)

   SDK は、aws.accessKeyId、aws.secretAccessKey、および aws.sessionToken JVM システムプロパティを検索します。

2. AWS アクセスキー (環境変数)

   SDK は、 AWS_ACCESS_KEY_IDと AWS_SECRET_ACCESS_KEY、およびAWS_SESSION_TOKEN環境変数から認証情報をロードしようとします。

3. ウェブ ID トークン

   SDK は環境変数 AWS_WEB_IDENTITY_TOKEN_FILEおよび AWS_ROLE_ARN (または JVM システムプロパティ aws.webIdentityTokenFileおよび) を検索しますaws.roleArn。トークン情報とロールに基づいて、SDK は一時的な認証情報を取得します。

4. 設定ファイルのプロファイル

   このステップでは、SDK はプロファイルに関連付けられた設定を使用します。デフォルトでは、SDK は共有 AWS config ファイルと credentials ファイルを使用しますが、AWS_CONFIG_FILE環境変数が設定されている場合、SDK はその値を使用します。AWS_PROFILE 環境変数 (または aws.profile JVM システムプロパティ) が設定されていない場合、SDK は「デフォルト」プロファイルを検索します。それ以外の場合は、AWS_PROFILE's値に一致するプロファイルを検索します。

   SDK は、前の段落で説明した設定に基づいてプロファイルを検索し、そこで定義された設定を使用します。SDK で見つかった設定に、さまざまな認証情報プロバイダーアプローチの設定が混在している場合、SDK は次の順序を使用します。

   a. AWS アクセスキー (設定ファイル) - SDK は、aws_access_key_id、aws_access_key_id、および の設定を使用しますaws_session_token。

   b. ロール設定の引き受け - SDK が role_arnおよび source_profileまたは credential_source設定を検出した場合、ロールの引き受けを試みます。SDK が source_profile設定を検出すると、別のプロファイルから認証情報をソースにして、 で指定されたロールの一時的な認証情報を受け取りますrole_arn。SDK が credential_source設定を検出すると、credential_source設定の値に応じて、Amazon ECS コンテナ、Amazon EC2 インスタンス、または環境変数から認証情報が取得されます。次に、これらの認証情報を使用して、ロールの一時的な認証情報を取得します。

      プロファイルには source_profile設定または credential_source設定を含める必要がありますが、両方を含めることはできません。

c. ウェブ ID トークン設定 - SDK が role_arnおよび web_identity_token_file設定を検出した場合、 role_arnとトークンに基づいて AWS リソースにアクセスするための一時的な認証情報を取得します。

d. SSO トークン設定 - SDK が sso_session、sso_account_id、 sso_role_name設定 (設定ファイル内のコンパニオンsso-sessionセクションとともに) を検出した場合、SDK は IAM Identity Center サービスから一時的な認証情報を取得します。

e. レガシー SSO 設定 - SDK が sso_start_url、sso_region、sso_account_id、および sso_role_name設定を検出した場合、SDK は IAM Identity Center サービスから一時的な認証情報を取得します。

f. プロセス設定 - SDK がcredential_process設定を見つけた場合、パス値を使用してプロセスを呼び出し、一時的な認証情報を取得します。

5. コンテナ認証情報

   SDK は、環境変数 AWS_CONTAINER_CREDENTIALS_RELATIVE_URIまたは AWS_CONTAINER_CREDENTIALS_FULL_URIと AWS_CONTAINER_AUTHORIZATION_TOKEN_FILEまたは を検索しますAWS_CONTAINER_AUTHORIZATION_TOKEN。これらの値を使用して、GET リクエストを通じて指定された HTTP エンドポイントから認証情報をロードします。

6. IMDS 認証情報

   SDK は、デフォルトまたは設定された HTTP エンドポイントでインスタンスメタデータサービスから認証情報を取得しようとします。SDK は IMDSv2 のみをサポートします。

この時点で認証情報がまだ解決されない場合、クライアントの作成は例外で失敗します。

> 注: 認証情報解決の順序の変更
>
> 上記の認証情報解決の順序は、 SDK for Kotlin の1.4.x+リリースでは最新です。1.4.0 リリース前は、項目番号 3 と 4 が切り替えられ、現在の 4a 項目は現在の 4f 項目に続きました。

明示的な認証情報プロバイダー

デフォルトのプロバイダーチェーンを使用する代わりに、SDK が使用する特定の認証情報プロバイダーまたはカスタムチェーン (CredentialsProviderChain) を指定できます。たとえば、

環境変数を使用してデフォルトの認証情報を設定する場合は、次のコードスニペットのように、EnvironmentCredentialsProviderをクライアントビルダーに提供します。

```
val ddb = DynamoDbClient {
    region = "us-east-1"
    credentialsProvider = EnvironmentCredentialsProvider()
}
```

> **Note**
>
> デフォルトのチェーンは認証情報をキャッシュしますが、スタンドアロンプロバイダーはキャッシュしません。CachedCredentialsProvider クラスを使用して任意の認証情報プロバイダーをラップすることで、すべての API コールで認証情報を不必要に取得しないようにできます。キャッシュされたプロバイダーは、現在の認証情報の有効期限が切れた場合にのみ新しい認証情報を取得します。

> **Note**
>
> CredentialsProvider インターフェイスを実装することで、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実装できます。

クライアントエンドポイントを設定する

が を AWS SDK for Kotlin 呼び出すとき AWS のサービス、最初のステップの 1 つは、リクエストをルーティングする場所を決定することです。このプロセスはエンドポイント解決と呼ばれます。

サービスクライアントを構築するときに、SDK のエンドポイント解決を設定できます。エンドポイント解決のデフォルト設定では通常問題ありませんが、デフォルト設定を変更する理由はいくつかあります。2 つの理由の例を次に示します。

- サービスのプレリリースバージョンまたはサービスのローカルデプロイにリクエストを行います。
- SDK でまだモデル化されていない特定のサービス機能へのアクセス。

> **Warning**
>
> エンドポイント解決は高度な SDK トピックです。デフォルト設定を変更すると、コードが破損するリスクがあります。デフォルト設定は、本番環境のほとんどのユーザーに適用されます。

カスタム設定

クライアントの構築時に使用できる 2 つのプロパティを使用して、サービスクライアントのエンドポイント解決をカスタマイズできます。

1. `endpointUrl: Url`
2. `endpointProvider: EndpointProvider`

`endpointUrl` を設定する

の値を設定endpointUrlして、サービスの「ベース」ホスト名を指定できます。ただし、この値はクライアントのEndpointProviderインスタンスにパラメータとして渡されるため、最終値ではありません。その後、EndpointProvider実装はその値を検査して変更し、最終的なエンドポイントを決定できます。

例えば、Amazon Simple Storage Service (Amazon S3) クライアントにendpointUrl値を指定してGetObjectオペレーションを実行すると、デフォルトのエンドポイントプロバイダー実装によってバケット名がホスト名値に挿入されます。

実際には、ユーザーはサービスの開発インスタンスまたはプレビューインスタンスを指すendpointUrl値を設定します。

`endpointProvider` を設定する

サービスクライアントのEndpointProvider実装によって、最終的なエンドポイント解決が決まります。次のコードブロックに示すEndpointProviderインターフェイスは、 resolveEndpointメソッドを公開します。

```
public fun interface EndpointProvider<T> {
    public suspend fun resolveEndpoint(params: T): Endpoint
}
```

サービスクライアントは、すべてのリクエストに対して resolveEndpointメソッドを呼び出します。サービスクライアントは、プロバイダーから返されたEndpoint値を使用しますが、それ以上の変更はありません。

EndpointProvider のプロパティ

resolveEndpoint メソッドは、エンドポイント解決で使用されるプロパティを含むサービス固有のEndpointParametersオブジェクトを受け入れます。

すべてのサービスには、次の基本プロパティが含まれます。

名前	型	説明
region	文字列	クライアントの AWS リージョン
endpoint	String	の値セットの文字列表現 endpointUrl
useFips	ブール値	クライアントの設定で FIPS エンドポイントが有効になっているかどうか
useDualStack	ブール値	クライアントの設定でデュアルスタックエンドポイントが有効になっているかどうか

サービスは、解決に必要な追加のプロパティを指定できます。例えば、Amazon S3 S3EndpointParametersにはバケット名といくつかの Amazon S3-specific機能設定が含まれています。たとえば、 forcePathStyleプロパティは仮想ホストアドレス指定を使用できるかどうかを決定します。

独自のプロバイダーを実装する場合、 の独自のインスタンスを作成する必要はありませんEndpointParameters。SDK は、各リクエストのプロパティを提供し、 の実装に渡しますresolveEndpoint。

endpointUrl、または endpointProvider

次の 2 つのステートメントは、同等のエンドポイント解決動作を持つクライアントを生成しないことを理解することが重要です。

```
// Use endpointUrl.
S3Client.fromEnvironment {
    endpointUrl = Url.parse("https://endpoint.example")
}

// Use endpointProvider.
S3Client.fromEnvironment {
    endpointProvider = object : S3EndpointProvider {
        override suspend fun resolveEndpoint(params: S3EndpointParameters): Endpoint =
 Endpoint("https://endpoint.example")
    }
}
```

endpointUrl プロパティを設定するステートメントは、 (デフォルト) プロバイダーに渡されるベース URL を指定します。これはエンドポイント解決の一部として変更できます。

を設定するステートメントは、 がS3Client使用する最終的な URL endpointProviderを指定します。

両方のプロパティを設定できますが、ほとんどの場合、カスタマイズが必要なプロパティのいずれかを指定します。一般的な SDK ユーザーとして、ほとんどの場合、 endpointUrl値を指定します。

Amazon S3 に関する注意事項

Amazon S3 は、バケット仮想ホスティングなどのカスタマイズされたエンドポイントのカスタマイズによってモデル化された機能の多くを備えた複雑なサービスです。仮想ホスティングは、バケット名がホスト名に挿入される Amazon S3 の機能です。

このため、Amazon S3 サービスクライアントのEndpointProvider実装を置き換えないことをお勧めします。解決動作を拡張する必要がある場合は、エンドポイントに関する追加の考慮事項とともにローカル開発スタックにリクエストを送信することで、デフォルトの実装をラップすることをお勧めします。次のendpointProvider例は、このアプローチの実装例を示しています。

例

`endpointUrl` の例

次のコードスニペットは、Amazon S3 クライアントの一般的なサービスエンドポイントを上書きする方法を示しています。

```
val client = S3Client.fromEnvironment {
    endpointUrl = Url.parse("https://custom-s3-endpoint.local")
    // EndpointProvider is left as the default.
}
```

`endpointProvider` の例

次のコードスニペットは、Amazon S3 のデフォルトの実装をラップするカスタムエンドポイントプロバイダーを提供する方法を示しています。

```
import aws.sdk.kotlin.services.s3.endpoints.DefaultS3EndpointProvider
import aws.sdk.kotlin.services.s3.endpoints.S3EndpointParameters
import aws.sdk.kotlin.services.s3.endpoints.S3EndpointProvider
import aws.smithy.kotlin.runtime.client.endpoints.Endpoint

public class CustomS3EndpointProvider : S3EndpointProvider {
    override suspend fun resolveEndpoint(params: S3EndpointParameters) =
        if (/* Input params indicate we must route another endpoint for whatever
 reason. */) {
            Endpoint(/* ... */)
        } else {
            // Fall back to the default resolution.
            DefaultS3EndpointProvider().resolveEndpoint(params)
        }
}
```

`endpointUrl` および `endpointProvider`

次のサンプルプログラムは、 endpointUrlと endpointProviderの設定間の相互作用を示しています。これは高度なユースケースです。

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.sdk.kotlin.services.s3.endpoints.DefaultS3EndpointProvider
import aws.sdk.kotlin.services.s3.endpoints.S3EndpointParameters
import aws.sdk.kotlin.services.s3.endpoints.S3EndpointProvider
```

```
import aws.smithy.kotlin.runtime.client.endpoints.Endpoint

fun main() = runBlocking {
    S3Client.fromEnvironment {
        endpointUrl = Url.parse("https://example.endpoint")
        endpointProvider = CustomS3EndpointProvider()
    }.use { s3 ->
        // ...
    }
}

class CustomS3EndpointProvider : S3EndpointProvider {
    override suspend fun resolveEndpoint(params: S3EndpointParameters) {
        // The resolved string value of the endpointUrl set in the client above is
 available here.
        println(params.endpoint)
        // ...
    }
}
```

HTTP

このセクションでは、 の HTTP 関連の設定について説明します AWS SDK for Kotlin。

トピック

- HTTP クライアント設定
- HTTP プロキシを使用する
- HTTP インターセプター
- 最小 TLS バージョンを強制する

HTTP クライアント設定

デフォルトでは、 は OkHttp に基づく HTTP クライアント AWS SDK for Kotlin を使用します。明示的に設定されたクライアントを指定することで、HTTP クライアントとその設定を上書きできます。

> **Warning**
>
> 使用する HTTP エンジンに関係なく、プロジェクト内の他の依存関係には、SDK に必要な特定のエンジンバージョンと競合する推移的な依存関係がある可能性があります。特

に、Spring Boot などのフレームワークは、OkHttp などの依存関係を管理し、SDK よりも古いバージョンに依存することが知られています。詳細についてはthe section called “依存関係の競合を解決するにはどうすればよいですか？”、「」を参照してください。

> **Note**
>
> デフォルトでは、各サービスクライアントは HTTP クライアントの独自のコピーを使用します。アプリケーションで複数のサービスを使用する場合は、単一の HTTP クライアントを構築し、すべてのサービスクライアントで共有できます。

基本的な設定

サービスクライアントを設定するときは、デフォルトのエンジンタイプを設定できます。SDK は、結果の HTTP クライアントエンジンを管理し、不要になったら自動的に閉じます。

次の例は、DynamoDB クライアントの初期化中の HTTP クライアントの設定を示しています。

インポート

```
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient
import kotlin.time.Duration.Companion.seconds
```

コード

```
DynamoDbClient {
    region = "us-east-2"
    httpClient {
        maxConcurrency = 64u
        connectTimeout = 10.seconds
    }
}.use { ddb ->

    // Perform some actions with Amazon DynamoDB.
}
```

HTTP エンジンタイプの指定

より高度なユースケースでは、エンジンタイプhttpClientを指定する追加のパラメータを に渡すことができます。これにより、そのエンジンタイプに固有の設定パラメータを設定できます。

次の例ではOkHttpEngine、 maxConcurrencyPerHostプロパティの設定に使用できる を指定します。

インポート

```
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient
import aws.smithy.kotlin.runtime.http.engine.okhttp.OkHttpEngine
```

コード

```
DynamoDbClient {
    region = "us-east-2"
    httpClient(OkHttpEngine) {  // The first parameter specifies the HTTP engine type.
        // The following parameter is generic HTTP configuration available in any
 engine type.
        maxConcurrency = 64u

            // The following parameter is OkHttp-specific configuration.
        maxConcurrencyPerHost = 32u
    }
}.use { ddb ->

    // Perform some actions with Amazon DynamoDB.
}
```

エンジンタイプに使用できる値は、OkHttpEngine、OkHttp4Engine、および ですCrtHttpEngine。

HTTP エンジンに固有の設定パラメータを使用するには、コンパイル時の依存関係としてエンジンを追加する必要があります。にはOkHttpEngine、Gradle を使用して次の依存関係を追加します。

(*X.Y.Z* リンクに移動して、利用可能な最新バージョンを確認できます)。

```
implementation(platform("aws.smithy.kotlin:bom:X.Y.Z"))
implementation("aws.smithy.kotlin:http-client-engine-okhttp")
```

にはCrtHttpEngine、次の依存関係を追加します。

```
implementation(platform("aws.smithy.kotlin:bom:X.Y.Z"))
implementation("aws.smithy.kotlin:http-client-engine-crt")
```

OkHttp4Engine を使用する

デフォルトの OkHttp4Engineを使用できない場合は、 を使用しますOkHttpEngine。smithy-kotlin GitHub リポジトリには、 の設定方法と使用方法に関する情報がありますOkHttp4Engine。

明示的な HTTP クライアントを使用する

明示的な HTTP クライアントを使用する場合、不要になったときの閉鎖を含め、その有効期間はお客様の責任となります。HTTP クライアントは、それを使用するサービスクライアントと同じ期間以上存続する必要があります。

次のコード例は、 がDynamoDbClientアクティブな間、HTTP クライアントを存続させるコードを示しています。use 関数は、HTTP クライアントが正しく閉じることを確認します。

インポート

```
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient
import aws.smithy.kotlin.runtime.http.engine.okhttp.OkHttpEngine
import kotlin.time.Duration.Companion.seconds
```

コード

```
OkHttpEngine {
    maxConcurrency = 64u
    connectTimeout = 10.seconds
}.use { okHttpClient ->

     DynamoDbClient {
        region = "us-east-2"
        httpClient = okHttpClient
    }.use { ddb ->
        {
            // Perform some actions with Amazon DynamoDB.
        }
    }
}
```

HTTP プロキシを使用する

を使用してプロキシサーバー AWS 経由で にアクセスするには AWS SDK for Kotlin、JVM システムプロパティまたは環境変数のいずれかを設定できます。両方が指定されている場合、JVM システムプロパティが優先されます。

JVM システムプロパティを使用する

SDK は JVM システムプロパティ `https.proxyHost`、`https.proxyPort`、および を検索しますhttp.nonProxyHosts。これらの一般的な JVM システムプロパティの詳細については、Java ドキュメントの「ネットワークとプロキシ」を参照してください。

```
java -Dhttps.proxyHost=10.15.20.25 -Dhttps.proxyPort=1234 -
Dhttp.nonProxyHosts=localhost|api.example.com MyApplication
```

環境変数を使用します。

SDK は、`https_proxy`、`http_proxy`、および `no_proxy`環境変数 (およびそれぞれの大文字バージョン) を検索します。

```
export http_proxy=http://10.15.20.25:1234
export https_proxy=http://10.15.20.25:5678
export no_proxy=localhost,api.example.com
```

EC2 インスタンスでプロキシを使用する

アタッチされた IAM ロールで起動された EC2 インスタンスでプロキシを設定する場合は、インスタンスメタデータへのアクセスに使用されるアドレスを必ず除外してください。これを行うには、`http.nonProxyHosts`JVM システムプロパティまたは`no_proxy`環境変数をインスタンスメタデータサービスの IP アドレス に設定します`169.254.169.254`。このアドレスは変化しません。

```
export no_proxy=169.254.169.254
```

HTTP インターセプター

インターセプターを使用して、API リクエストとレスポンスの実行にフックできます。インターセプターは、SDK が記述したコードを呼び出してリクエスト/レスポンスライフサイクルに動作を挿入するオープンエンドメカニズムです。これにより、処理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。

次の例は、再試行ループを入力する前にすべての送信リクエストにヘッダーを追加するシンプルなインターセプターを示しています。

```
class AddHeader(
    private val key: String,
    private val value: String
) : HttpInterceptor {
    override suspend fun modifyBeforeRetryLoop(context:
 ProtocolRequestInterceptorContext<Any, HttpRequest>): HttpRequest {
        val httpReqBuilder = context.protocolRequest.toBuilder()
        httpReqBuilder.headers[key] = value
        return httpReqBuilder.build()
    }
}
```

詳細と使用可能な傍受フックについては、「インターセプターインターフェイス」を参照してください。

インターセプターの登録

インターセプターは、サービスクライアントを構築するとき、または特定の一連のオペレーションの設定を上書きするときに登録します。

すべてのサービスクライアントオペレーションのインターセプター

次のコードは、ビルダーのインターセプタープロパティにAddHeaderインスタンスを追加します。この追加により、再試行ループを入力する前に、すべてのオペレーションに x-foo-versionヘッダーが追加されます。

```
val s3 = S3Client.fromEnvironment {
    interceptors += AddHeader("x-foo-version", "1.0")
}

// All service operations invoked using 's3' will have the header appended.
s3.listBuckets { ... }
s3.listObjectsV2 { ... }
```

特定のオペレーションのみのインターセプター

withConfig 拡張機能を使用すると、任意のサービスクライアントの 1 つ以上のオペレーションのサービスクライアント設定を上書きできます。この機能を使用すると、オペレーションのサブセットに追加のインターセプターを登録できます。

次の の例では、 use拡張機能内のオペレーションのs3インスタンスの設定を上書きします。で呼び出されるオペレーションs3Scopedには、 ヘッダーx-foo-versionと x-bar-versionヘッダーの両方が含まれます。

```
// 's3' instance created in the previous code snippet.
s3.withConfig {
    interceptors += AddHeader("x-bar-version", "3.7")
}.use { s3Scoped ->
    // All service operations invoked using 's3Scoped' trigger interceptors
    // that were registered when the client was created and any added in the
    // withConfig { ... } extension.
}
```

最小 TLS バージョンを強制する

では AWS SDK for Kotlin、サービスエンドポイントに接続するときに最小 TLS バージョンを設定できます。SDK にはさまざまな設定オプションがあります。優先順位が最も高い順に、オプションは次のとおりです。

- HTTP エンジンを明示的に設定する
- sdk.minTls JVM システムプロパティを設定する
- SDK_MIN_TLS 環境変数を設定する

HTTP エンジンを設定する

サービスクライアントにデフォルト以外の HTTP エンジンを指定すると、tlsContext.minVersionフィールドを設定できます。

次の例では、少なくとも TLS v1.2 を使用するように HTTP エンジンとそれを使用するサービスクライアントを設定します。

```
DynamoDbClient {
    region = "us-east-2"
    httpClient {
        tlsContext {
            minVersion = TlsVersion.TLS_1_2
        }
    }
}.use { ddb ->
```

```
    // Perform some actions with Amazon DynamoDB.
}
```

`sdk.minTls` JVM システムプロパティを設定する

`sdk.minTls` JVM システムプロパティを設定できます。システムプロパティセットを使用してアプリケーションを起動すると、 によって AWS SDK for Kotlin 構築されたすべての HTTP エンジンは、デフォルトで指定された最小 TLS バージョンを使用します。ただし、これは HTTP エンジン設定で明示的に上書きできます。使用できる値は次のとおりです。

- TLS_1_0
- TLS_1_1
- TLS_1_2
- TLS_1_3

`SDK_MIN_TLS` 環境変数を設定する

`SDK_MIN_TLS` 環境変数を設定できます。環境変数を設定した状態でアプリケーションを起動すると、 によって構築されたすべての HTTP エンジンは、別のオプションによって上書きされない限り、指定された最小 TLS バージョン AWS SDK for Kotlin を使用します。

使用できる値は次のとおりです。

- TLS_1_0
- TLS_1_1
- TLS_1_2
- TLS_1_3

再試行

を呼び出すと、予期しない例外が返される AWS のサービス ことがあります。呼び出しを再試行すると、スロットリングや一時的なエラーなど、特定のタイプのエラーが成功する可能性があります。

このページでは、 を使用して自動再試行を設定する方法について説明します AWS SDK for Kotlin。

デフォルトの再試行設定

デフォルトでは、すべてのサービスクライアントは標準の再試行戦略で自動的に設定されます。デフォルト設定では、最大 3 回失敗する呼び出しを試行します (最初の試行と 2 回の再試行) 。各呼び出し間の介入遅延は、再試行ストームを回避するために、エクスポネンシャルバックオフとランダムジッターで設定されます。この設定は、ほとんどのユースケースで機能しますが、高スループットシステムなどの状況では適さない場合があります。

SDK は、再試行可能なエラーに対してのみ再試行を試みます。再試行可能なエラーの例としては、ソケットタイムアウト、サービス側のスロットリング、同時実行または楽観的なロック障害、一時的なサービスエラーなどがあります。パラメータの欠落または無効、認証/セキュリティエラー、設定ミスの例外は再試行可能と見なされません。

最大試行回数、遅延とバックオフ、トークンバケットの設定を設定することで、標準の再試行戦略をカスタマイズできます。

最大試行回数

クライアントの構築中に `retryStrategy` DSL ブロックのデフォルトの最大試行回数 (3) をカスタマイズできます。

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy {
        maxAttempts = 5
    }
}
```

前のスニペットで示した DynamoDB サービスクライアントでは、SDK は最大 5 回失敗する API コールを試行します (最初の試行と 4 回の再試行) 。

次のスニペットに示すように、最大試行回数を 1 に設定することで、自動再試行を完全に無効にできます。

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy {
        maxAttempts = 1  // The SDK makes no retries.
    }
}
```

遅延とバックオフ

再試行が必要な場合、デフォルトの再試行戦略は後続の試行を行う前に待機します。最初の再試行の遅延は小さいですが、後の再試行では指数関数的に増加します。最大遅延量は、大きくなりすぎないように制限されます。

最後に、すべての試行間の遅延にランダムジッターが適用されます。ジッターは、再試行ストームを引き起こす可能性のある大規模なフリートの影響を軽減するのに役立ちます。(エクスポネンシャルバックオフとジッターの詳細については、このAWS アーキテクチャブログの投稿を参照してください) 。

遅延パラメータは delayProviderDSL ブロックで設定できます。

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy {
        delayProvider {
            initialDelay = 100.milliseconds
            maxBackoff = 5.seconds
        }
    }
}
```

前のスニペットに示されている設定では、クライアントは最初の再試行を最大 100 ミリ秒遅延します。再試行間の最大時間は 5 秒です。

遅延とバックオフの調整には、次のパラメータを使用できます。

パラメータ	デフォルト値	説明
initialDelay	10 ミリ秒	最初の再試行の最大遅延時間。ジッターを適用すると、実際の遅延量が少なくなる可能性があります。
jitter	1.0 (フルジッター)	計算された遅延をランダムに減らす最大振幅。デフォルト値の 1.0 は、計算された遅延を最大 100% (0 など) まで減らすことができることを意味

パラメータ	デフォルト値	説明
		します。値 0.5 は、計算された遅延を最大半分に減らすことができることを意味します。したがって、最大遅延 10 ミリ秒を 5 ミリ秒から 10 ミリ秒の範囲で短縮できます。値 0.0 は、ジッターが適用されないことを意味します。 **Important** #ジッター設定は高度な機能です。この動作のカスタマイズは通常お勧めしません。
`maxBackoff`	20 秒	試行に適用される最大遅延時間。この値を設定すると、その後の試行の間に発生する指数関数的な増加が制限され、計算された最大値が大きすぎることが防止されます。このパラメータは、ジッターが適用されるまでの計算遅延を制限します。適用されると、ジッターによって遅延がさらに減少する可能性があります。

パラメータ	デフォルト値	説明
`scaleFactor`	1.5	後続の最大遅延が増加する指数ベース。たとえば、`initialDelay` が 10 ミリ秒、`scaleFactor` が 1.5 の場合、次の最大遅延が計算されます。 • 再試行 1: 10 ミリ秒 × 1.50 = 10 ミリ秒 • 再試行 2: 10 ミリ秒 × 1.51 = 15 ミリ秒 • 再試行 3: 10 ミリ秒 × 1.52 = 22.5 ミリ秒 • 再試行 4: 10 ミリ秒 × 1.53 = 33.75 ミリ秒 ジッターを適用すると、各遅延の実際の量が少なくなる可能性があります。

トークンバケットを再試行する

デフォルトのトークンバケット設定を調整することで、標準の再試行戦略の動作をさらに変更できます。再試行トークンバケットは、成功する可能性が低い再試行や、タイムアウトやスロットリングの失敗など、解決に時間がかかる再試行を減らすのに役立ちます。

> **Important**
>
> トークンバケット設定は高度な機能です。この動作のカスタマイズは通常お勧めしません。

再試行するたびに (オプションで最初の試行を含む) 、トークンバケットから一部の容量が減少します。減少する量は、試行のタイプによって異なります。例えば、一時的なエラーを再試行すると安く

なりますが、タイムアウトまたはスロットリングエラーを再試行するとコストが高くなる可能性があります。

試行が成功すると、バケットに容量が返されます。バケットは最大容量を超えて増分したり、ゼロ未満に減らしたりすることはできません。

useCircuitBreakerMode 設定の値に応じて、 は容量を 0 未満に減らそうとすると、次のいずれかの結果になります。

- 設定が TRUE の場合、例外がスローされます。たとえば、再試行回数が多すぎて、それ以上の再試行が成功する可能性が低い場合などです。
- 設定が FALSE の場合、遅延が発生します。たとえば、バケットが再び十分な容量になるまで遅延します。

トークンバケットパラメータは tokenBucketDSL ブロックで設定できます。

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy {
        tokenBucket {
            maxCapacity = 100
            refillUnitsPerSecond = 2
        }
    }
}
```

再試行トークンバケットのチューニングには、次のパラメータを使用できます。

パラメータ	デフォルト値	説明
initialTryCost	0	最初の試行でバケットから減らす量。デフォルト値の 0 は、容量が減少しないため、最初の試行が停止または遅延しないことを意味します。
initialTrySuccessIncrement	1	最初の試行が成功したときの容量の増加量。

パラメータ	デフォルト値	説明
maxCapacity	500	トークンバケットの最大容量。使用可能なトークンの数はこの数を超えることはできません。
refillUnitsPerSecond	0	1 秒ごとにバケットに再追加された容量。値 0 は、容量が自動的に再追加されないことを意味します。(たとえば、試行が成功すると容量が増加します)。値 0 は TRUE useCircuitBreakerMode である必要があります。
retryCost	5	一時的な障害後に試行するためにバケットから減少する量。試行が成功すると、同じ量がバケットに再増分されます。
timeoutRetryCost	10	タイムアウトまたはスロットリングの失敗後に試行するためにバケットから減少する量。試行が成功すると、同じ量がバケットに再増分されます。

パラメータ	デフォルト値	説明
`useCircuitBreakerMode`	正	容量を減らそうとすると、バケットの容量がゼロを下回る動作を決定します。TRUE の場合、トークンバケットは再試行容量が存在しないことを示す例外をスローします。FALSE の場合、トークンバケットは十分な容量が補充されるまで試行を遅らせます。

アダプティブ再試行

標準再試行戦略の代わりに、アダプティブ再試行戦略は、スロットリングエラーを最小限に抑えるための理想的なリクエストレートを求める高度なアプローチです。

> **Important**
>
> アダプティブ再試行は高度な再試行モードです。この再試行戦略を使用することは通常お勧めしません。

アダプティブ再試行には、標準再試行のすべての機能が含まれます。これにより、スロットリングされていないリクエストと比較したスロットリングされたリクエストのレートを測定するクライアント側のレートリミッターが追加されます。また、トラフィックが安全な帯域幅内にとどまるように制限し、スロットリングエラーが発生するのが理想的です。

レートは、サービス条件やトラフィックパターンの変化にリアルタイムで適応し、それに応じてトラフィックレートを増減する可能性があります。クリティカルなことに、レートリミッターはトラフィックの多いシナリオでの最初の試行を遅らせる可能性があります。

`retryStrategy` メソッドに追加のパラメータを指定して、アダプティブ再試行戦略を選択します。レートリミッターパラメータは `rateLimiter`DSL ブロックで設定できます。

```
val dynamoDb = DynamoDbClient.fromEnvironment {
```

```
    retryStrategy(AdaptiveRetryStrategy) {
        maxAttempts = 10
        rateLimiter {
            minFillRate = 1.0
            smoothing = 0.75
        }
    }
}
```

> **Note**
>
> アダプティブ再試行戦略では、クライアントが 1 つのリソース (1 つの DynamoDB テーブルまたは 1 つの Amazon S3 バケットなど) に対して動作することを前提としています。1 つのクライアントを複数のリソースに使用すると、1 つのリソースに関連付けられたスロットリングまたは停止により、クライアントが他のすべてのリソースにアクセスするときにレイテンシーが増加し、障害が発生します。アダプティブ再試行戦略を使用する場合は、リソースごとに 1 つのクライアントを使用することをお勧めします。

オブザーバビリティ

オブザーバビリティは、システムが出力するデータからシステムの現在の状態を推測できる範囲です。出力されるデータは、一般的にテレメトリと呼ばれます。

は、メトリクス、トレース、ログの 3 つの一般的なテレメトリシグナルをすべて提供 AWS SDK for Kotlin できます。をワイヤアップ`TelemetryProvider`して、テレメトリデータをオブザーバビリティバックエンド (AWS X-Rayや Amazon CloudWatch など) に送信し、それに対応できます。

デフォルトでは、SDK ではログ記録のみが有効になり、他のテレメトリ信号は無効になります。このトピックでは、テレメトリ出力を有効にして設定する方法について説明します。

> **Important**
>
> `TelemetryProvider` は現在、使用するにはオプトインする必要がある実験的な API です。

を設定する TelemetryProvider

アプリケーションTelemetryProvider内の は、すべてのサービスクライアントまたは個々のクライアントに対してグローバルに設定できます。次の例では、仮定getConfiguredProvider()関数を使用して TelemetryProvider API オペレーションを示します。the section called “テレメトリプロバイダー” このセクションでは、 SDK が提供する実装について説明します。プロバイダーがサポートされていない場合は、独自のサポートを実装するか、GitHub で機能リクエストを開くことができます。

デフォルトのグローバルテレメトリプロバイダーを設定する

デフォルトでは、すべてのサービスクライアントはグローバルに利用可能なテレメトリプロバイダーの使用を試みます。これにより、プロバイダーを 1 回設定でき、すべてのクライアントがそれを使用します。これは、サービスクライアントをインスタンス化する前に 1 回だけ行う必要があります。

グローバルテレメトリプロバイダーを使用するには、まずプロジェクトの依存関係を更新して、次の Gradle スニペットに示すようにテレメトリのデフォルトモジュールを追加します。

(*X.Y.Z* リンクに移動して、利用可能な最新バージョンを確認できます)。

```
dependencies {
    implementation(platform("aws.smithy.kotlin:bom:X.Y.Z"))
    implementation("aws.smithy.kotlin:telemetry-defaults")
    ...
}
```

次に、次のコードに示すように、サービスクライアントを作成する前にグローバルテレメトリプロバイダーを設定します。

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.smithy.kotlin.runtime.telemetry.GlobalTelemetryProvider
import kotlinx.coroutines.runBlocking

fun main() = runBlocking {
    val myTelemetryProvider = getConfiguredProvider()
    GlobalTelemetryProvider.set(myTelemetryProvider)

    S3Client.fromEnvironment().use { s3 ->
        …
```

```
    }
}

fun getConfiguredProvider(): TelemetryProvider {
    TODO("TODO - configure a provider")
}
```

特定のサービスクライアントのテレメトリプロバイダーを設定する

個々のサービスクライアントは、特定のテレメトリプロバイダー (グローバルプロバイダーを除く) で設定できます。以下の例ではこれを示しています。

```
import aws.sdk.kotlin.services.s3.S3Client
import kotlinx.coroutines.runBlocking

fun main() = runBlocking {
    S3Client.fromEnvironment{
        telemetryProvider = getConfiguredProvider()
    }.use { s3 ->
        ...
    }
}

fun getConfiguredProvider(): TelemetryProvider {
    TODO("TODO - configure a provider")
}
```

メトリクス

次の表に、SDK が出力するテレメトリメトリクスを示します。メトリクスを観測可能にするようにテレメトリプロバイダーを設定します。

どのようなメトリクスが出力されますか？

メトリクス名	単位	タイプ	属性	説明
smithy.client.call.duration	s	ヒストグラム	rpc.service、rpc.method	全体的な通話時間 (再試行を含む)
smithy.client.call.attempts	{attempt}	MonotonicCounter	rpc.service、rpc.method	個々のオペレーションの試行回数

メトリクス名	単位	タイプ	属性	説明
smithy.client.call.errors	{error}	Monoton Counter	rpc.servi ce、rpc.me thod、 exception .type	オペレーションのエラーの数
smithy.client.call.attempt_ duration	s	ヒスト グラム	rpc.servi ce、rpc.method	サービスへの接続、リクエストの送信、HTTP ステータスコードとヘッダーの取得にかかる時間 (キューに入れられた送信待機時間を含む)
smithy.client.call.resolve_ endpoint_ duration	s	ヒスト グラム	rpc.servi ce、rpc.method	リクエストのエンドポイント (DNS ではなくエンドポイントリゾルバー) の解決にかかる時間
smithy.client.call.serialization_dur ation	s	ヒスト グラム	rpc.servi ce、rpc.method	メッセージ本文のシリアル化にかかる時間
smithy.cl ient.call.deserial ization_duration	s	ヒスト グラム	rpc.servi ce、rpc.method	メッセージ本文の逆シリアル化にかかる時間
smithy.cl ient.call.auth.sig ning_duration	s	ヒスト グラム	rpc.servi ce、rpc.me thod、auth .scheme_id	リクエストの署名にかかる時間
smithy.client.call.auth.res olve_iden tity_duration	s	ヒスト グラム	rpc.servi ce、rpc.me thod、auth .scheme_id	ID プロバイダーから ID (AWS 認証情報やベアラートークンなど) を取得するのにかかる時間

メトリクス名	単位	タイプ	属性	説明
smithy.client.http.connections.acquire_duration	s	ヒストグラム		接続の取得リクエストにかかる時間
smithy.client.http.connections.limit	{接続}	〔非同期]UpDownCount		HTTP クライアントで許可/設定されるオープン接続の最大数
smithy.client.http.connections.usage	{接続}	〔非同期]UpDownCount	状態: アイドル | 取得済み	接続プールの現在の状態
smithy.client.http.connections.uptime	s	ヒストグラム		接続が開いている時間
smithy.client.http.requests.usage	{request}	〔非同期]UpDownCount	状態: キューに入っている | 処理中	HTTP クライアントリクエストの同時実行の現在の状態
smithy.client.http.requests.queued_duration	s	ヒストグラム		リクエストがキューに入れられ、HTTP クライアントによって実行されるのを待った時間
smithy.client.http.bytes_sent	方法	MonotonCounter	server.address	HTTP クライアントによって送信された総バイト数
smithy.client.http.bytes_received	方法	MonotonCounter	server.address	HTTP クライアントが受信した総バイト数

列の説明は次のとおりです。

- メトリクス名 – 出力されたメトリクスの名前。

- Units – メトリクスの測定単位。単位は、UCUM の大文字と小文字を区別する (「c/s」) 表記で指定します。
- Type – メトリクスのキャプチャに使用される計測器のタイプ。
- 説明 – メトリクスが測定する内容の説明。
- 属性 – メトリクスで出力される属性 (ディメンション) のセット。

ログ記録

は、SLF4J 互換ロガーをテレメトリプロバイダーLoggerProviderのデフォルトとして AWS SDK for Kotlin 設定します。抽象化レイヤーである SLF4J では、実行時に複数のロギングシステムのいずれかを使用できます。サポートされているログ記録システムには、Java ログ記録 APIs、Log4j 2、および Logback が含まれます。

> **Warning**
>
> デバッグの目的でのみワイヤログ記録を使用することをお勧めします。(ワイヤログ記録については以下で説明します)。E メールアドレス、セキュリティトークン、API キー、パスワード、 AWS Secrets Manager シークレットなどの機密データをログに記録する可能性があるため、本番環境ではオフにします。ワイヤログ記録は、HTTPS 呼び出しであっても、暗号化なしでリクエストまたはレスポンス全体をログに記録します。
> 大規模なリクエスト (Amazon S3 へのファイルのアップロードなど) やレスポンスの場合、詳細なワイヤログ記録も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。

Log4j 2 ログ記録設定の例

SLF4J互換のログライブラリを使用できますが、この例では Log4j 2 を使用して JVM プログラム内の SDK からのログ出力を有効にします。

Gradle の依存関係

(*X.Y.Z* リンクに移動して、利用可能な最新バージョンを確認できます)。

```
implementation("org.apache.logging.log4j:log4j-slf4j2-impl:X.Y.Z")
```

Log4j 2 設定ファイル

resources ディレクトリlog4j2.xmlに という名前のファイルを作成します (例: <project-dir>/src/main/resources) 。次の XML 設定を ファイルに追加します。

```
<Configuration status="ERROR">
    <Appenders>
        <Console name="Out">
            <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} %-5p %c:%L %X - %encode{%m}
{CRLF}%n"/>
        </Console>
    </Appenders>
    <Loggers>
        <Root level="info">
            <AppenderRef ref="Out"/>
        </Root>
    </Loggers>
</Configuration>
```

この設定には、MDC (マッピングされた診断コンテキスト) ログ記録を有効にする pattern 属性の%X指定子が含まれます。

SDK は、オペレーションごとに次の MDC 要素を追加します。

rpc

呼び出した RPC の名前。例: S3.GetObject。

sdkInvocationId

オペレーションのためにサービスクライアントによって割り当てられた一意の ID。ID は、1 回のオペレーションの呼び出しに関連するすべてのログイベントを関連付けます。

ワイヤレベルのメッセージのログモードを指定する

デフォルトでは、 AWS SDK for Kotlin は、API リクエストとレスポンスからの機密データが含まれている可能性があるため、ワイヤレベルのメッセージをログに記録しません。ただし、デバッグの目的でこの詳細レベルが必要になる場合があります。

Kotlin SDK を使用すると、コードでログモードを設定するか、環境設定を使用して以下のデバッグメッセージングを有効にできます。

- HTTP リクエスト

- HTTP レスポンス

ログモードはビットフィールドでバックアップされ、各ビットはフラグ (モード) で、値は付加的です。1 つのリクエストモードと 1 つのレスポンスモードを組み合わせることができます。

コードでログモードを設定する

追加のログ記録をオプトインするには、サービスクライアントを構築するときに logModeプロパティを設定します。

次の例は、リクエスト (本文あり) とレスポンス (本文なし) のログ記録を有効にする方法を示しています。

```
import aws.smithy.kotlin.runtime.client.LogMode

// ...

val client = DynamoDbClient {
    // ...
    logMode = LogMode.LogRequestWithBody + LogMode.LogResponse
}
```

サービスクライアントの構築中に設定されたログモード値は、環境から設定されたログモード値を上書きします。

環境からログモードを設定する

コードで明示的に設定されていないすべてのサービスクライアントに対してログモードをグローバルに設定するには、次のいずれかを使用します。

- JVM システムプロパティ: sdk.logMode
- 環境変数: SDK_LOG_MODE

以下の大文字と小文字を区別しない値を使用できます。

- LogRequest
- LogRequestWithBody
- LogResponse
- LogResponseWithBody

環境の設定を使用して結合ログモードを作成するには、値をパイプ (|) 記号で区切ります。

たとえば、次の例では、前の例と同じログモードを設定します。

```
# Environment variable.
export SDK_LOG_MODE=LogRequestWithBody|LogResponse
```

```
# JVM system property.
java -Dsdk.logMode=LogRequestWithBody|LogResponse ...
```

> **Note**
>
> また、互換性のある SLF4J ロガーを設定し、ログレベルを DEBUG に設定して、ワイヤレベルのログ記録を有効にする必要があります。

テレメトリプロバイダー

SDK は現在、プロバイダーとして OpenTelemetry (OTel) をサポートしています。SDK は、将来的に追加のテレメトリプロバイダーを提供する場合があります。

トピック

- OpenTelemetry ベースのテレメトリプロバイダーを設定する

OpenTelemetry ベースのテレメトリプロバイダーを設定する

SDK for Kotlin は、OpenTelemetry がサポートするTelemetryProviderインターフェイスの実装を提供します。

前提条件

次の Gradle スニペットに示すように、プロジェクトの依存関係を更新して OpenTelemetry プロバイダーを追加します。*X.Y.Z* リンクに移動して、利用可能な最新バージョンを確認できます。

```
dependencies {
    implementation(platform("aws.smithy.kotlin:bom:X.Y.Z"))
    implementation(platform("io.opentelemetry.instrumentation:opentelemetry-
instrumentation-bom:X.Y.Z"))
```

```
    implementation("aws.smithy.kotlin:telemetry-provider-otel")

    // OPTIONAL: If you use log4j, the following entry enables the ability to export
 logs through OTel.
    runtimeOnly("io.opentelemetry.instrumentation:opentelemetry-log4j-appender-2.17")
}
```

SDK を設定

次のコードは、OpenTelemetry テレメトリプロバイダーを使用してサービスクライアントを設定します。

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.smithy.kotlin.runtime.telemetry.otel.OpenTelemetryProvider
import io.opentelemetry.api.GlobalOpenTelemetry
import kotlinx.coroutines.runBlocking

fun main() = runBlocking {
    val otelProvider = OpenTelemetryProvider(GlobalOpenTelemetry.get())

    S3Client.fromEnvironment().use { s3 ->
        telemetryProvider = otelProvider
        …
    }
}
```

> **Note**
>
> OpenTelemetry SDK の設定方法については、このガイドの範囲外です。OpenTelemetry Java ドキュメントには、手動、Java エージェントによる自動、または (オプション) コレクターなど、さまざまなアプローチに関する設定情報が含まれています。

リソース

OpenTelemetry の使用を開始するには、以下のリソースが役立ちます。

- AWS Distro for OpenTelemetry - AWS OTeL Distro ホームページ
- aws-otel-java-instrumentation - AWS Distro for OpenTelemetry Java Instrumentation Library
- aws-otel-lambda AWS マネージド OpenTelemetry Lambda レイヤー

- aws-otel-collector - AWS Distro for OpenTelemetry Collector
- AWS オブザーバビリティのベストプラクティス - 固有のオブザーバビリティの一般的なベストプラクティス AWS

サービスクライアント設定を上書きする

サービスクライアントが作成されると、サービスクライアントはすべてのオペレーションに固定設定を使用します。ただし、1 つ以上の特定のオペレーションの設定を上書きする必要がある場合があります。

各サービスクライアントにはwithConfig拡張機能があり、既存の設定のコピーを変更できます。withConfig 拡張機能は、設定を変更した新しいサービスクライアントを返します。元のクライアントは独立して存在し、元の設定を使用します。

次の例は、2 つのオペレーションを呼び出すS3Clientインスタンスの作成を示しています。

```
val s3 = S3Client.fromEnvironment {
    logMode = LogMode.LogRequest
    region = "us-west-2"
    // ...other configuration settings...
}

s3.listBuckets { ... }
s3.listObjectsV2 { ... }
```

次のスニペットは、単一のlistObjectV2オペレーションの設定を上書きする方法を示しています。

```
s3.withConfig {
    region = "eu-central-1"
}.use { overriddenS3 ->
    overriddenS3.listObjectsV2 { ... }
}
```

s3 クライアントの オペレーション呼び出しは、クライアントの作成時に指定された元の設定を使用します。その設定には、 リージョンus-west-2 regionのリクエストログ記録と が含まれます。

overriddenS3 クライアントのlistObjectsV2呼び出しは、 リージョンを除き、元のs3クライアントと同じ設定を使用します。リージョンは現在 ですeu-central-1。

オーバーライドされたクライアントのライフサイクル

前の例では、s3クライアントとoverriddenS3クライアントは互いに独立しています。オペレーションは、開いている限り、どちらのクライアントでも呼び出すことができます。各 は個別の設定を使用しますが、基盤となるリソース (HTTP エンジンなど) も上書きされない限り共有できます。

上書きされた設定でクライアントを閉じ、元のクライアントを個別に閉じます。元のクライアントを閉じる前または後に、上書きされた設定でクライアントを閉じることができます。設定が上書きされたクライアントを長期間使用する必要がない限り、そのライフサイクルを useメソッドでラップすることをお勧めします。use メソッドは、例外が発生した場合にクライアントを閉じるようにします。

クライアント間で共有されるリソース

を使用してサービスクライアントを作成するとwithConfig、元のクライアントとリソースを共有する場合があります。対照的に、fromEnvironment を使用してクライアントを作成するか、明示的に設定すると、クライアントは独立したリソースを使用します。HTTP エンジンや認証情報プロバイダーなどのリソースは、 withConfigブロックで上書きされない限り共有されます。

各クライアントのライフサイクルは独立しているため、共有リソースは最後のクライアントが閉鎖されるまで開いたままで使用できます。したがって、オーバーライドされたサービスクライアントが不要になったら、それらを閉じることが重要です。これにより、共有リソースが開いたままになり、メモリ、接続、CPU サイクルなどのシステムリソースを消費するのを防ぐことができます。

次の例は、共有リソースと独立リソースの両方を示しています。

s3 および overriddenS3クライアントは、キャッシュ設定を含め、同じ認証情報プロバイダーインスタンスを共有します。キャッシュされた値がs3クライアントによって行われた呼び出しからまだ最新である場合、認証情報をoverriddenS3再利用して行われた呼び出し。

HTTP エンジンは 2 つのクライアント間で共有されません。withConfig 呼び出しで上書きされたため、各クライアントには独立した HTTP エンジンがあります。

```
val s3 = S3Client.fromEnvironment {
    region = "us-west-2"
    credentialsProvider = CachedCredentialsProvider(CredentialsProviderChain(...))
    httpClientEngine = OkHttpEngine { ... }
}

s3.listBuckets { ... }
```

```
s3.withConfig {
    httpClientEngine = CrtHttpEngine { ... }
}.use { overriddenS3 ->
    overriddenS3.listObjectsV2 { ... }
}
```

SDK を使用する

このセクションでは、 を使用するために必要な基本情報を提供します AWS SDK for Kotlin。

トピック

- リクエストを発行する
- コルーチン
- ストリーミング操作
- ページ分割
- ウェーター
- エラー処理
- リクエストの事前署名
- トラブルシューティングに関するよくある質問
- でのモック AWS SDK for Kotlin

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 にリクエストを行います AWS のサービス。 AWS SDK for Kotlin は、タイプセーフなビルダーパターンに従ってドメイン固有の言語 (DSLs) を提供し、リクエストを作成します。リクエストのネストされた構造には、DSLs からもアクセスできます。

次の例は、Amazon DynamoDB createTable オペレーション入力を作成する方法を示しています。

```
val ddb = DynamoDbClient.fromEnvironment()

val req = CreateTableRequest {
    tableName = name
    keySchema = listOf(
        KeySchemaElement {
            attributeName = "year"
            keyType = KeyType.Hash
        },
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
```

```
        }
    )

    attributeDefinitions = listOf(
        AttributeDefinition {
            attributeName = "year"
            attributeType = ScalarAttributeType.N
        },
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }
    )

    // You can configure the `provisionedThroughput` member
    // by using the `ProvisionedThroughput.Builder` directly:
    provisionedThroughput {
        readCapacityUnits = 10
        writeCapacityUnits = 10
    }
}

val resp = ddb.createTable(req)
```

サービスインターフェイスの DSL オーバーロード

サービスクライアントインターフェイスの各非ストリーミングオペレーションには DSL オーバーロードがあるため、個別のリクエストを作成する必要はありません。

オーバーロードされた関数を使用して Amazon Simple Storage Service (Amazon S3) バケットを作成する例：

```
s3Client.createBucket {    // this: CreateBucketRequest.Builder
    bucket = newBucketName
}
```

これは次に相当します:

```
val request = CreateBucketRequest {    // this: CreateBucketRequest.Builder
    bucket = newBucketName
}
```

```
s3client.createBucket(request)
```

入力を必要としないリクエスト

必要な入力がないオペレーションは、リクエストオブジェクトを渡すことなく呼び出すことができます。これは多くの場合、Amazon S3 API オペレーションなどのリストタイプのオペレーションで可能です。 `listBuckets`

たとえば、次の 3 つのステートメントは同等です。

```
s3Client.listBuckets(ListBucketsRequest {
  // Construct the request object directly.
})
s3Client.listBuckets {
  // DSL builder without explicitly setting any arguments.
}
s3Client.listBuckets()
```

コルーチン

AWS SDK for Kotlin はデフォルトで非同期です。SDK for Kotlin は、コルーチンから呼び出されるすべてのオペレーションに suspend関数を使用します。

コルーチンの詳細なガイドについては、公式の Kotlin ドキュメントを参照してください。

同時リクエストの実行

非同期コルーチンビルダーを使用して、結果を重視する同時リクエストを起動できます。 は、後で結果を提供する promise を表す、軽量でノンブロッキングの未来を表す Deferred asyncを返します。

結果に関心がない場合 (オペレーションが完了した場合のみ)、起動コルーチンビルダーを使用できます。 launchは概念的に に似ていますasync。違いは、起動がジョブを返し、結果の値は持たず、 は asyncを返すことですDeferred。

以下は、headObject オペレーションを使用して Amazon S3 への同時リクエストを行い、2 つのキーのコンテンツサイズを取得する例です。

```
import kotlinx.coroutines.async
import kotlinx.coroutines.runBlocking
```

```
import kotlin.system.measureTimeMillis
import aws.sdk.kotlin.services.s3.S3Client


fun main(): Unit = runBlocking {

    val s3 = S3Client { region = "us-east-2" }

    val myBucket = "<your-bucket-name-here>"
    val key1 = "<your-object-key-here>"
    val key2 = "<your-second-object-key-here>"

    val resp1 = async {
        s3.headObject{
            bucket = myBucket
            key = key1
        }
    }

    val resp2 = async {
        s3.headObject{
            bucket = myBucket
            key = key2
        }
    }


    val elapsed = measureTimeMillis {
        val totalContentSize = resp1.await().contentLength +
 resp2.await().contentLength
        println("content length of $key1 + $key2 = $totalContentSize")
    }

    println("requests completed in $elapsed ms")

}
```

ブロックリクエストの作成

コルーチンを使用せず、別のスレッドモデルを実装する既存のコードからサービス呼び出しを行うには、runBlocking コルーチンビルダーを使用できます。別のスレッドモデルの例としては、Java の従来のエグゼキュター/未来アプローチの使用があります。Java と Kotlin のコードまたはライブラリをブレンドする場合は、このアプローチを使用する必要がある場合があります。

名前が示すように、このrunBlockingビルダーは新しいコルーチンを起動し、完了するまで現在のスレッドをブロックします。

> ⚠ Warning
>
> runBlocking 通常、コルーチンからは使用しないでください。これは、通常のブロックコードを一時停止スタイルで記述されたライブラリ (主要な関数やテストなど) にブリッジするように設計されています。

ストリーミング操作

では AWS SDK for Kotlin、バイナリデータ (ストリーム) は ByteStreamタイプとして表されます。これはバイトの抽象読み取り専用ストリームです。

ストリーミングレスポンス

バイナリストリームを使用したレスポンス (Amazon Simple Storage Service (Amazon S3) GetObject API オペレーションなど) は、他の方法とは異なる方法で処理されます。これらのメソッドは、レスポンスを直接返すのではなく、レスポンスを処理する Lambda 関数を使用します。これにより、関数へのレスポンスの範囲が制限され、呼び出し元と SDK ランタイムの両方のライフタイム管理が簡素化されます。

Lambda 関数が戻ると、基盤となる HTTP 接続などのリソースが解放されます。(Lambda が戻った後は にアクセスByteStreamしないでください。また、閉鎖から渡さないでください。) 呼び出しの結果は、Lambda が返す結果です。

次のコード例は、レスポンスを処理する lambda パラメータを受け取る getObject 関数を示しています。

```
val s3Client = S3Client.fromEnvironment()
val req = GetObjectRequest { ... }

val path = Paths.get("/tmp/download.txt")

// S3Client.getObject has the following signature:
// suspend fun <T> getObject(input: GetObjectRequest, block: suspend
 (GetObjectResponse) -> T): T

val contentSize = s3Client.getObject(req) { resp ->
```

```
    // resp is valid until the end of the block.
    // Do not attempt to store or process the stream after the block returns.

    // resp.body is of type ByteStream.
    val rc = resp.body?.writeToFile(path)
    rc
}
println("wrote $contentSize bytes to $path")
```

ByteStream タイプには、一般的な使用方法として次の拡張子があります。

- ByteStream.writeToFile(file: File): Long
- ByteStream.writeToFile(path: Path): Long
- ByteStream.toByteArray(): ByteArray
- ByteStream.decodeToString(): String

これらはすべて aws.smithy.kotlin.runtime.contentパッケージで定義されています。

ストリーミングリクエスト

を指定するにはByteStream、次のような便利な方法もあります。

- ByteStream.fromFile(file: File)
- File.asByteStream(): ByteStream
- Path.asByteStream(): ByteStream
- ByteStream.fromBytes(bytes: ByteArray)
- ByteStream.fromString(str: String)

これらはすべて aws.smithy.kotlin.runtime.contentパッケージで定義されています。

次のコード例は、PutObjectRequest の作成時に本文プロパティを提供するByteStream便利なメソッドの使用を示しています。

```
val req = PutObjectRequest {
    ...
    body = ByteStream.fromFile(file)
    // body = ByteStream.fromBytes(byteArray)
    // body = ByteStream.fromString("string")
```

```
    // etc
}
```

ページ分割

多くの AWS オペレーションは、ペイロードが大きすぎて 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 Kotlin には、結果を自動的にページ分割するサービスクライアントインターフェイスの拡張機能が含まれています。結果を処理するコードを記述するだけで済みます。

ページ分割は Flow<T> として公開されるため、非同期コレクション (map、、 など) に対して Kotlin のイディオマティック変換を活用できますfiltertake。例外は透過的であるため、エラー処理は通常の API コールのように感じられ、キャンセルはコルーチンの一般的な協調的なキャンセルに従います。詳細については、公式ガイドのhttps://kotlinlang.org/docs/flow.html「フローとフロー例外」を参照してください。

> **Note**
>
> 次の例ではAmazon S3を使用しています。ただし、概念は、ページ分割された APIs。すべてのページ分割拡張機能は、 aws.sdk.kotlin.<service>.paginatorsパッケージ (などaws.sdk.kotlin.dynamodb.paginators) で定義されます。

次のコード例は、listObjectsV2Paginated 関数呼び出しからのページ分割されたレスポンスを処理する方法を示しています。

インポート

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.sdk.kotlin.services.s3.paginators.listObjectsV2Paginated
import kotlinx.coroutines.flow.*
```

コード

```
val s3 = S3Client.fromEnvironment()
val req = ListObjectsV2Request {
    bucket = "<my-bucket>"
    maxKeys = 1
```

```
}

s3.listObjectsV2Paginated(req)  // Flow<ListObjectsV2Response>
    .transform { it.contents?.forEach { obj -> emit(obj) } }
    .collect { obj ->
        println("key: ${obj.key}; size: ${obj.size}")
    }
```

ウェーター

ウェーターは、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移行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クライアント側の抽象化です。これは、Amazon Simple Storage Service (Amazon S3) などの結果整合性のあるサービスや、Amazon EC2 などのリソースを非同期的に作成するサービスを使用する場合の一般的なタスクです。

リソースのステータスを継続的にポーリングするロジックを記述すると、面倒でエラーが発生しやすくなります。ウェーターの目標は、この責任をカスタマーコードから に移行することです。この は AWS SDK for Kotlin、 AWS オペレーションのタイミングの側面について深い知識を持っています。

> **Note**
>
> 次の例ではAmazon S3を使用しています。ただし、概念は、1 つ以上のウェーターが定義されている AWS のサービス でも同じです。すべての拡張機能はaws.sdk.kotlin.<service>.waitersパッケージで定義されます (などaws.sdk.kotlin.dynamodb.waiters) 。また、標準の命名規則 () に従いますwaitUntil<Condition>。

次のコード例は、ポーリングロジックの記述を回避できるウェーター関数の使用を示しています。

インポート

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.sdk.kotlin.services.s3.waiters.waitUntilBucketExists
```

コード

```
val s3 = S3Client.fromEnvironment()
```

```
// This initiates creating an S3 bucket and potentially returns before the bucket
 exists.
s3.createBucket { bucket = "my-bucket" }

// When this function returns, the bucket either exists or an exception
// is thrown.
s3.waitUntilBucketExists { bucket = "my-bucket" }

// The bucket now exists.
```

> **Note**
>
> 各待機メソッドは、目的の条件に達することに対応する最終レスポンスで取得するために使用できるOutcomeインスタンスを返します。結果には、目的の状態に到達しようとした回数などの詳細も含まれています。

エラー処理

が例外を AWS SDK for Kotlin スローする方法とタイミングを理解することは、 SDK を使用して高品質のアプリケーションを構築する上で重要です。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。

サービス例外

最も一般的な例外は でAwsServiceException、サービス固有の例外 (などS3Exception) はすべて継承されます。この例外は、 AWS のサービスからのエラーレスポンスを表します。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と、Amazon EC2 はエラーレスポンスを返します。エラーレスポンスの詳細は、スローAwsServiceExceptionされる に含まれています。

が発生するとAwsServiceException、リクエストは に正常に送信されました AWS のサービスが、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。

クライアントの例外

ClientException は、 にリクエストを送信しようとしたとき、 AWS またはレスポンスを AWS SDK for Kotlin 解析しようとしたときに、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。は通常ClientException、 よりも重要度が高くAwsServiceException、クライアントが

へのサービス呼び出しを処理できないという重大な問題があることを示します AWS のサービス。例えば、 は、サービスからのレスポンスの解析に失敗ClientExceptionした場合、 を AWS SDK for Kotlin スローします。

エラーメタデータ

すべてのサービス例外とクライアント例外には、 sdkErrorMetadataプロパティがあります。これは、エラーに関する追加の詳細を取得するために使用できる型付きプロパティバッグです。

AwsErrorMetadata タイプには、いくつかの事前定義された拡張機能が直接存在します。これには、以下が含まれますが、これらに限定されません。

- sdkErrorMetadata.requestId – 一意のリクエスト ID
- sdkErrorMetadata.errorMessage – 人間が読み取れるメッセージ (通常は と一致しますがException.message、例外がサービスに不明な場合は詳細情報が含まれる場合があります)
- sdkErrorMetadata.protocolResponse – raw プロトコルレスポンス

次の例は、エラーメタデータへのアクセスを示しています。

```
try {
    s3Client.listBuckets { ... }
} catch (ex: S3Exception) {
    val awsRequestId = ex.sdkErrorMetadata.requestId
    val httpResp = ex.sdkErrorMetadata.protocolResponse as? HttpResponse

    println("requestId was: $awsRequestId")
    println("http status code was: ${httpResp?.status}")
}
```

リクエストの事前署名

一部の AWS API オペレーションのリクエストに事前署名することで、別の発信者が自分の認証情報を提示せずに後でリクエストを使用できるようになります。

例えば、Alice が Amazon Simple Storage Service (Amazon S3) オブジェクトにアクセスできるが、Bob とオブジェクトアクセスを一時的に共有したいとします。Alice は、署名付きGetObjectリクエストを生成して Bob と共有し、Alice の認証情報にアクセスすることなくオブジェクトをダウンロードできます。

署名の基本

SDK for Kotlin は、サービスクライアントで拡張メソッドを提供し、リクエストに事前署名します。すべての署名付きリクエストには、署名付きリクエストが有効である期間を表す期間が必要です。期間が終了すると、署名付きリクエストは期限切れになり、実行されると認証エラーが発生します。

次のコードは、Amazon S3 の署名付き`GetObject`リクエストを作成する例を示しています。リクエストは作成後 24 時間有効です。

```
val s3 = S3Client.fromEnvironment()

val unsignedRequest = GetObjectRequest {
    bucket = "foo"
    key = "bar"
}

val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)
```

`presignGetObject` 拡張メソッドは `HttpRequest` オブジェクトを返します。リクエストオブジェクトには、オペレーションを呼び出すことができる署名付き URL が含まれています。別の発信者は、別のコードベースまたはプログラミング言語環境で URL (またはリクエスト全体) を使用できます。

署名付きリクエストを作成したら、HTTP クライアントを使用してリクエストを呼び出します。HTTP GET リクエストを呼び出す API は、HTTP クライアントによって異なります。次の例では、Kotlin `URL.readText`メソッドを使用しています。

```
val objectContents = URL(presignedRequest.url.toString()).readText()
println(objectContents)
```

高度な事前署名設定

SDK では、リクエストに事前署名できる各メソッドにオーバーロードがあり、特定の署名者の実装や詳細な署名パラメータなどの高度な設定オプションを提供するために使用できます。

次の例は、CRT 署名者バリアントを使用し、将来の署名日を指定する Amazon S3 `GetObject`リクエストを示しています。

```
val s3 = S3Client.fromEnvironment()
```

```
val unsignedRequest = GetObjectRequest {
    bucket = "foo"
    key = "bar"
}

val presignedRequest = s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
    signingDate = Instant.now() + 24.hours
    expiresAfter = 8.hours
}
```

返される署名付きリクエストは 24 時間前に転送され、それまでは有効ではありません。有効期限は 8 時間後です。

POST リクエストと PUT リクエストの事前署名

事前署名可能なオペレーションの多くは URL のみを必要とし、HTTP GET リクエストとして実行する必要があります。ただし、一部のオペレーションは本文を受け取り、場合によってはヘッダーとともに HTTP POST または HTTP PUT リクエストとして実行する必要があります。これらのリクエストの事前署名は GET リクエストの事前署名と同じですが、署名付きリクエストの呼び出しはより複雑です。

S3 `PutObject`リクエストに事前署名する例を次に示します。

```
val s3 = S3Client.fromEnvironment()

val unsignedRequest = PutObjectRequest {
    bucket = "foo"
    key = "bar"
}

val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)
```

返される `HttpRequest`のメソッド値は `HttpMethod.PUT`で、HTTP リクエストの今後の呼び出しに含める必要がある URL とヘッダーが含まれています。このリクエストは、別のコードベースまたはプログラミング言語環境で実行できる発信者に渡すことができます。

署名付き POST または PUT リクエストを作成したら、HTTP クライアントを使用してリクエストを呼び出します。POST または PUT リクエスト URL を呼び出す API は、使用する HTTP クライアントによって異なります。次の例では、OkHttp HTTP クライアントを使用し、 を含む本文が含まれていますHello world。

```
val putRequest = Request
    .Builder()
    .url(presignedRequest.url.toString())
    .apply {
        presignedRequest.headers.forEach { key, values ->
            header(key, values.joinToString(", "))
        }
    }
    .put("Hello world".toRequestBody())
    .build()

val response = okHttp.newCall(putRequest).execute()
```

SDK が事前署名できるオペレーション

SDK for Kotlin は現在、関連付けられた HTTP メソッドで呼び出す必要がある次の API オペレーションの事前署名をサポートしています。

AWS のサービス	Operation	SDK 拡張機能メソッド	HTTP メソッドを使用する
Amazon S3	GetObject	presignGetObject	HTTP GET
Amazon S3	PutObject	presignPutObject	HTTP PUT
Amazon S3	UploadPart	presignUploadPart	HTTP PUT
AWS Security Token Service	GetCallerIdentity	presignGetCallerIdentity	HTTP POST
Amazon Polly	SynthesizeSpeech	presignSynthesizeSpeech	HTTP POST

トラブルシューティングに関するよくある質問

AWS SDK for Kotlin 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いくつかの問題が発生する可能性があります。根本原因を明らかにしてエラーを解決するには、次の提案を参考にしてください。

「接続が閉じられた」問題を修正するにはどうすればよいですか？

次のいずれかのタイプなどの例外として、「接続が閉じられた」問題が発生する場合があります。

- `IOException: unexpected end of stream on <URL>`
- `EOFException: \n not found: limit=0`
- `HttpException: AWS_ERROR_HTTP_CONNECTION_CLOSED: The connection has closed or is closing.; crtErrorCode=2058; HttpErrorCode(CONNECTION_CLOSED)`

これらの例外は、SDK からサービスへの TCP 接続が予期せず閉じられたか、リセットされたことを示します。接続は、ホスト、 AWS サービス、または NAT ゲートウェイ、プロキシ、ロードバランサーなどの仲介者によって閉じられた可能性があります。

これらのタイプの例外は自動的に再試行されますが、ログ記録設定によっては SDK ログに表示される場合があります。例外がコードにスローされた場合、アクティブな再試行戦略が、最大試行回数や再試行トークンバケットなどの設定された制限を使い果たしたことを示します。再試行戦略の詳細については、このガイドのthe section called “再試行”「」セクションを参照してください。the section called “最大試行回数に達する前に例外がスローされるのはなぜですか？” トピックも参照してください。

最大試行回数に達する前に例外がスローされるのはなぜですか？

再試行が予想されたが、代わりにスローされた例外が表示される場合があります。このような状況では、次の手順が問題の解決に役立つ場合があります。

- 例外が再試行可能であることを確認します。例として、不正な形式のサービスリクエスト、アクセス許可の欠如、存在しないリソースを示す例外など、一部の例外は再試行できません。SDK は、これらの種類の例外を自動的に再試行しません。から継承する例外をキャッチする場合は`SdkBaseException`、ブールプロパティをチェックして`SdkBaseException.sdkErrorMetadata.isRetryable`、SDK が例外を再試行可能と判断したかどうかを確認できます。
- 例外がコードにスローされていることを確認します。一部の例外はログメッセージに情報として表示されますが、実際にはコードにスローされません。例えば、スロットリングエラーなどの再試行可能な例外は、SDK が複数のbackoff-and-retryサイクルを自動的に実行するため、ログに記録される場合があります。SDK オペレーションの呼び出しは、設定された再試行設定で処理されなかった場合にのみ例外をスローします。

- 設定された再試行設定を確認します。再試行戦略と再試行ポリシーの詳細については、このガイドのthe section called “再試行”「」セクションを参照してください。コードが想定した設定または自動デフォルトを使用していることを確認します。
- 再試行設定の調整を検討してください。前の項目を検証しても問題が解決しない場合は、再試行設定の調整を検討してください。
  - 最大試行回数を増やします。デフォルトでは、 オペレーションの最大試行回数は 3 です。これが十分ではなく、デフォルト設定で例外が発生している場合は、クライアント設定で `retryStrategy.maxAttempts`プロパティを増やすことを検討してください。詳細については「the section called “最大試行回数”」を参照してください。
  - 遅延設定を増やします。一部の例外は、基盤となる条件が解決される前に、あまりにも迅速に再試行される可能性があります。その場合は、クライアント設定で `retryStrategy.delayProvider.initialDelay`または `retryStrategy.delayProvider.maxBackoff`プロパティを増やすことを検討してください。詳細については「the section called “遅延とバックオフ”」を参照してください。
  - サーキットブレーカーモードを無効にします。SDK は、デフォルトで各サービスクライアントのトークンのバケットを維持します。SDK がリクエストを試行し、再試行可能な例外で失敗すると、トークン数は減少し、リクエストが成功すると、トークン数は増加します。

    デフォルトでは、このトークンバケットが残り 0 トークンに達すると、回路が壊れます。回路が切断されると、SDK は再試行を無効にし、最初の試行で失敗した現在および後続のリクエストは直ちに例外をスローします。最初の試行が成功すると、SDK はトークンバケットに十分な容量を返した後、再試行を再度有効にします。この動作は意図的なものであり、サービス停止やサービス復旧中の再試行ストームを防ぐように設計されています。

    SDK が設定された最大試行回数まで再試行し続ける場合は、クライアント設定で `retryStrategy.tokenBucket.useCircuitBreakerMode`プロパティを false に設定して、サーキットブレーカーモードを無効にすることを検討してください。このプロパティを false に設定すると、SDK クライアントはトークンバケットが十分な容量に達するまで待機します。残りのトークンが 0 個ある場合、例外につながる可能性のある再試行は中止されません。

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

これらのエラーは、依存関係の欠落または競合によって最も一般的に発生します。詳細については「the section called “依存関係の競合を解決するにはどうすればよいですか？”」を参照してください。

`NoClassDefFoundError` の が表示されます。 `okhttp3/coroutines/ExecuteAsyncKt`

これは、特に OkHttp の依存関係の問題を示します。詳細については「the section called “アプリケーションでの OkHttp バージョンの競合の解決”」を参照してください。

依存関係の競合を解決するにはどうすればよいですか？

を使用する場合は AWS SDK for Kotlin、特定の AWS およびサードパーティーの依存関係が正しく機能する必要があります。これらの依存関係が実行時に欠落しているか、予期しないバージョンがある場合、 `NoSuchMethodError`や などの エラーが表示されることがありますNoClassDefFoundError。これらの依存関係の問題は、通常 2 つのグループに分類されます。

- SDK/Smithy 依存関係の競合
- サードパーティーの依存関係の競合

Kotlin アプリケーションを構築するときは、Gradle を使用して依存関係を管理する可能性があります。SDK サービスクライアントへの依存関係をプロジェクトに追加すると、必要なすべての関連依存関係が自動的に含まれます。ただし、アプリケーションに他の依存関係がある場合、SDK で必要な依存関係と競合する可能性があります。例えば、SDK は、アプリケーションが使用する可能性のある一般的な HTTP クライアントである OkHttp に依存しています。これらの競合を見つけるために、Gradle はプロジェクトの依存関係を一覧表示する便利なタスクを提供します。

```
./gradlew dependencies
```

依存関係の競合が発生した場合は、アクションを実行する必要がある場合があります。特定のバージョンの依存関係またはシャドウ依存関係をローカル名前空間に指定できます。Gradle 依存関係解決は、Gradle ユーザーマニュアルの以下のセクションで説明する複雑なトピックです。

- 依存関係の解決について
- 依存関係の制約と競合の解決
- 依存関係バージョンの調整

プロジェクトの SDK と Smithy の依存関係の管理

SDK を使用する場合は、そのモジュールは通常、バージョン番号が一致する他の SDK モジュールに依存することに注意してください。たとえば、 `aws.sdk.kotlin:s3:1.2.3`は に依存し`ws.sdk.kotlin:aws-http:1.2.3`、 は に依存し`aws.sdk.kotlin:aws-core:1.2.3`ます。

SDK モジュールは、特定の Smithy モジュールバージョンも使用します。Smithy モジュールバージョンは SDK バージョン番号と同期しませんが、SDK の想定バージョンと一致する必要があります。たとえば、 `aws.sdk.kotlin:s3:1.2.3`は に依存し`aws.smithy.kotlin:serde:1.1.1`、は に依存し`aws.smithy.kotlin:runtime-core:1.1.1`ます。

依存関係の競合を回避するには、すべての SDK 依存関係を一緒にアップグレードし、明示的な Smithy 依存関係に対して同じ操作を行います。Gradle バージョンカタログを使用してバージョンを同期させ、SDK と Smithy バージョン間のマッピングにおける推測を排除することを検討してください。

SDK/Smithy モジュールのマイナーバージョン更新には、バージョニングポリシーで説明されているように、重大な変更が含まれる場合があることに注意してください。マイナーバージョン間でアップグレードする場合は、変更ログを注意深く確認し、ランタイム動作を徹底的にテストします。

アプリケーションでの OkHttp バージョンの競合の解決

OkHttp は、SDK が JVM でデフォルトで使用する一般的な HTTP エンジンです。アプリケーションには、別の OkHttp バージョンを取り込む他の依存関係やフレームワークが含まれる場合があります。これにより、 `okhttp/coroutines/ExecuteAsyncKt` や など、`okhttp3`名前空間内の`NoClassDefFoundError`クラスの が発生する可能性があります`okhttp3/ConnectionListener`。この場合、通常、新しいバージョンを選択して競合を解決する必要があります。これらの競合の原因を追跡しやすくするために、Gradle は便利なタスクを提供します。以下を実行して、すべての依存関係を一覧表示できます。

```
./gradlew dependencies
```

たとえば、SDK が OkHttp に依存し`5.0.0-alpha.14`、Spring Boot などの別の依存関係が OkHttp に依存している場合は`4.12.0`、 を使用する必要があります`5.0.0-alpha.14 version`。これを行うには、Gradle の `constraints`ブロックを使用します。

```
dependencies {
    constraints {
        implementation("com.squareup.okhttp3:okhttp:4.12.0")
    }
```

```
}
```

または、OkHttp 4.x を使用する必要がある場合、SDK は を提供しますOkHttp4Engine。Gradle を設定してコードOkHttp4Engineで使用する方法については、 ドキュメントを参照してください。

でのモック AWS SDK for Kotlin

開発者は、複数のフレームワークを使用して、 を使用したテストでモックを実行できます AWS SDK for Kotlin。このトピックでは、一部のフレームワークに必要な追加の設定や特別な考慮事項について説明します。

MockK

MockK を使用してモジュール全体の拡張機能をモックする場合は、追加の設定を行う必要があります。SDK for Kotlin では、ページネーター、ウェーター、およびプレシグナーが拡張関数の例であるため、動作をモックするときは追加の設定が必要です。

モックを設定するmockkStatic("<MODULE_CLASS_NAME>")前に を呼び出す必要があります。原則として、モジュールクラス名は次のとおりです。

- ページネーター: aws.sdk.kotlin.services.<service>.paginators.PaginatorsKt
- ウェーター: aws.sdk.kotlin.services.<service>.waiters.WaitersKt
- 署名者: aws.sdk.kotlin.services.<service>.presigners.PresignersKt

たとえば、ページネーター拡張関数listBucketsPaginatedで
あるモックを含む次のテストでは、 を追加しま
すmockkStatic("aws.sdk.kotlin.services.s3.paginators.PaginatorsKt")。

```
    @Test
    fun testPaginatedListBuckets() = runTest {

        mockkStatic("aws.sdk.kotlin.services.s3.paginators.PaginatorsKt")
        val s3Client: S3Client = mockk()
        val s3BucketLister = S3BucketLister(s3Client)

        val expectedBuckets = listOf(
            Bucket { name = "bucket1" },
            Bucket { name = "bucket2" }
        )
```

```
        val response = ListBucketsResponse { buckets = expectedBuckets }
        coEvery { s3Client.listBucketsPaginated() } returns flowOf(response)

        val result = s3BucketLister.getAllBucketNames()

        assertEquals(listOf("bucket1", "bucket2"), result)
    }
```

を使用しない場合mockkStatic、次のエラーが表示されます。

```
Missing mocked calls inside every { ... } block: make sure the object inside the block
 is a mock
io.mockk.MockKException: Missing mocked calls inside every { ... } block: make sure the
 object inside the block is a mock
    at
 io.mockk.impl.recording.states.StubbingState.checkMissingCalls(StubbingState.kt:14)
    at io.mockk.impl.recording.states.StubbingState.recordingDone(StubbingState.kt:8)
    at io.mockk.impl.recording.CommonCallRecorder.done(CommonCallRecorder.kt:47)
    at io.mockk.impl.eval.RecordedBlockEvaluator.record(RecordedBlockEvaluator.kt:63)
    at io.mockk.impl.eval.EveryBlockEvaluator.every(EveryBlockEvaluator.kt:30)
    at io.mockk.MockKDsl.internalCoEvery(API.kt:100)
    at io.mockk.MockKKt.coEvery(MockK.kt:174)
```

を使用しない署名者拡張関数の場合mockkStatic、以下が表示されることがあります。

```
key is bound to the URI and must not be null
java.lang.IllegalArgumentException: key is bound to the URI and must not be null
    at
 aws.sdk.kotlin.services.s3.serde.GetObjectOperationSerializer.serialize(GetObjectOperationSeri
    at
 aws.sdk.kotlin.services.s3.presigners.PresignersKt.presignGetObject(Presigners.kt:49)
    at aws.sdk.kotlin.services.s3.presigners.PresignersKt.presignGetObject
$default(Presigners.kt:40)
    at aws.sdk.kotlin.services.s3.presigners.PresignersKt.presignGetObject-
exY8QGI(Presigners.kt:30)
```

すべてのアーティファクトの例

テスト対象のコード

```
import aws.sdk.kotlin.services.s3.S3Client
```

```
import aws.sdk.kotlin.services.s3.paginators.listBucketsPaginated
import kotlinx.coroutines.flow.filter
import kotlinx.coroutines.flow.toList
import kotlinx.coroutines.flow.transform
import kotlinx.coroutines.runBlocking
import org.slf4j.Logger
import org.slf4j.LoggerFactory

fun main() {
    val logger: Logger = LoggerFactory.getLogger(::main.javaClass)

    // Create an S3Client
    S3Client { region = "us-east-1" }.use { s3Client ->
        // Create service instance
        val bucketLister = S3BucketLister(s3Client)
        // Since getAllBucketNames is a suspend function, you'll need to run it in a
 coroutine scope
        runBlocking {
            val bucketNames = bucketLister.getAllBucketNames()
            logger.info("Found buckets: $bucketNames")
        }
    }
}

class S3BucketLister(private val s3Client: S3Client) {
    suspend fun getAllBucketNames(): List<String> {
        return s3Client.listBucketsPaginated()
            .transform { response ->
                response.buckets?.forEach { bucket ->
                    emit(bucket.name ?: "")
                }
            }
            .filter { it.isNotEmpty() }
            .toList()
    }
}
```

テストクラス

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.sdk.kotlin.services.s3.model.Bucket
import aws.sdk.kotlin.services.s3.model.ListBucketsResponse
import aws.sdk.kotlin.services.s3.paginators.listBucketsPaginated
```

```
import io.mockk.coEvery
import io.mockk.mockk
import io.mockk.mockkStatic
import kotlinx.coroutines.flow.flowOf
import kotlinx.coroutines.test.runTest
import org.junit.jupiter.api.Assertions.assertEquals
import org.junit.jupiter.api.Test

class S3BucketListerTest {

    @Test
    fun testPaginatedListBuckets() = runTest {

        mockkStatic("aws.sdk.kotlin.services.s3.paginators.PaginatorsKt")
        val s3Client: S3Client = mockk()
        val s3BucketLister = S3BucketLister(s3Client)

        val expectedBuckets = listOf(
            Bucket { name = "bucket1" },
            Bucket { name = "bucket2" }
        )

        val response = ListBucketsResponse { buckets = expectedBuckets }
        coEvery { s3Client.listBucketsPaginated() } returns flowOf(response)

        val result = s3BucketLister.getAllBucketNames()

        assertEquals(listOf("bucket1", "bucket2"), result)
    }
}
```

build.gradle.kts

```
plugins {
    kotlin("jvm") version "2.1.20"
    application
}

group = "org.example"
version = "1.0-SNAPSHOT"

repositories {
    mavenCentral()
```

```
}

dependencies {
    implementation(platform(awssdk.bom))
    implementation(platform("org.apache.logging.log4j:log4j-bom:2.24.3"))

    implementation(awssdk.services.s3)
    implementation("org.apache.logging.log4j:log4j-slf4j2-impl")

    // Testing Dependencies
    testImplementation(platform("org.junit:junit-bom:5.11.0"))
    testImplementation("org.junit.jupiter:junit-jupiter")
    testImplementation("org.jetbrains.kotlinx:kotlinx-coroutines-test:1.7.3")
    testImplementation("io.mockk:mockk:1.14.0")
}

tasks.test {
    useJUnitPlatform()
}

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(21)
    }
}

application {
    mainClass = "org.example.S3BucketService"
}
```

settings.gradle.kts

```
plugins {
    id("org.gradle.toolchains.foojay-resolver-convention") version "0.10.0"
}
rootProject.name = "mockK-static"

dependencyResolutionManagement {
    repositories {
        mavenCentral()
    }

    versionCatalogs {
```

```
        create("awssdk") {
            from("aws.sdk.kotlin:version-catalog:1.4.69")
        }
    }
}
```

AWS のサービス を使用して を操作する AWS SDK for Kotlin

この章では、 SDK for Kotlin AWS のサービス を使用して を操作する方法について説明します。

目次

- を使用して Amazon S3 を操作する AWS SDK for Kotlin
  - チェックサムによるデータ整合性保護
    - オブジェクトのアップロード
      - 事前に計算されたチェックサム値を使用してください。
      - マルチパートアップロード
    - オブジェクトのダウンロード
      - 非同期検証
  - SDK for Kotlin を使用して Amazon S3 マルチリージョンアクセスポイントを操作する
    - マルチリージョンアクセスポイントの使用
    - オブジェクトとマルチリージョンアクセスポイントの操作
- を使用して DynamoDB を操作する AWS SDK for Kotlin
  - AWS アカウントベースのエンドポイントを使用する
  - DynamoDB マッパーを使用してクラスを DynamoDB 項目にマッピングする (開発者プレビュー)
    - DynamoDB Mapper の使用を開始する
      - 依存関係を追加する
      - マッパーを作成して使用する
      - クラス注釈を使用してスキーマを定義する
      - 呼び出しオペレーション
        - ページ分割されたレスポンスを使用する
    - DynamoDB マッパーを設定する
      - インターセプターを使用する
        - リクエストパイプラインを理解する
        - フック
          - 読み取り専用フック

        - フックの変更
        - 実行順序
        - 設定例
- 注釈からスキーマを生成する
    - プラグインを適用する
    - プラグインを設定する
    - クラスに注釈を付ける
        - クラス注釈
        - プロパティ注釈
    - カスタム項目コンバーターを定義する
- スキーマを手動で定義する
    - コードでスキーマを定義する
- DynamoDB Mapper で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のスキーマを定義する
    - オペレーションでセカンダリインデックスを使用する
- 式を使用する
    - オペレーションで式を使用する
    - DSL コンポーネント
        - 属性
        - 等価と不等価
        - 範囲とセット
        - ブールロジック
        - 関数とプロパティ
        - ソートキーフィルター

を使用して Amazon S3 を操作する AWS SDK for Kotlin

Kotlin SDK の Amazon Simple Storage Service への主なインターフェイスは S3Client です。SDK の他のサービスクライアントS3Clientと同様に を使用して、Amazon S3 にリクエストを行います。

S3 で Kotlin SDK を使用するのに役立つリソースは次のとおりです。

- S3 の Kotlin SDK API リファレンス。
- S3 サービスユーザーガイド」および「サービス API リファレンス」。

以下のトピックでは、S3 で動作する一部の Kotlin SDK APIs のガイド付きコード例を示します。

トピック

- チェックサムによるデータ整合性保護
- SDK for Kotlin を使用して Amazon S3 マルチリージョンアクセスポイントを操作する

チェックサムによるデータ整合性保護

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 のチェックサムの詳細については、サポートされているアルゴリズムを含む Amazon Simple Storage Service ユーザーガイドを参照してください。

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アルゴリズムのいずれかを使用して、事前に計算されたチェックサム値を指定することもできます。

> **Note**
>
> のバージョン 1.4.0 以降 AWS SDK for Kotlin、SDK はアップロードのCRC32チェックサムを自動的に計算することでデフォルトの整合性保護を提供します。事前計算されたチェックサム値を指定しない場合、または SDK がチェックサムの計算に使用するアルゴリズムを指定しない場合、SDK はこのチェックサムを計算します。
> SDK には、外部で設定できるデータ整合性保護のグローバル設定も用意されています。詳細については、AWS SDKs」を参照してください。

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

リクエストパラメータを指定した `putObject` 関数を使用して、SDK for Kotlin を使用して Amazon S3 にオブジェクトをアップロードします。リクエストデータ型には、チェックサム計算を有効にする `checksumAlgorithm` プロパティがあります。

次のコードスニペットは、CRC32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、CRC32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。

```
val request = PutObjectRequest {
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    checksumAlgorithm = ChecksumAlgorithm.CRC32
}
```

リクエストでチェックサムアルゴリズムを指定しない場合、チェックサムの動作は、次の表に示すように、使用する SDK のバージョンによって異なります。

チェックサムアルゴリズムが指定されていない場合のチェックサムの動作

Kotlin SDK バージョン	チェックサムの動作
1.4.0 より前	SDK は CRC ベースのチェックサムを自動的に計算せず、リクエストで指定します。
1.4.0 以降	SDK はCRC32アルゴリズムを使用してチェックサムを計算し、リクエストで提供します。Amazon S3 は、独自のCRC32チェックサムを計算して転送の整合性を検証し、SDK が提供するチェックサムと比較します。チェックサムが一致した場合、チェックサムは オブジェクトとともに保存されます。

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、SHA256 チェックサムが事前に計算されたリクエストを示しています。

```
val request = PutObjectRequest {
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    body = ByteStream.fromFile(File("file_to_upload.txt"))
    checksumAlgorithm = ChecksumAlgorithm.SHA256
    checksumSha256 = "cfb6d06da6e6f51c22ae3e549e33959dbb754db75a93665b8b579605464ce299"
}
```

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。

CreateMultipartUpload リクエストと各UploadPartリクエストでチェックサムアルゴリズムを指定する必要があります。最後のステップとして、CompleteMultipartUpload の各部分のチェックサムを指定する必要があります。次の例では、チェックサムアルゴリズムを指定してマルチパートアップロードを作成する方法を示しています。

```
val multipartUpload = s3.createMultipartUpload {
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    checksumAlgorithm = ChecksumAlgorithm.Sha1
}

val partFilesToUpload = listOf("data-part1.csv", "data-part2.csv", "data-part3.csv")

val completedParts = partFilesToUpload
    .mapIndexed { i, fileName ->
        val uploadPartResponse = s3.uploadPart {
            bucket = "amzn-s3-demo-bucket"
            key = "key"
            body = ByteStream.fromFile(File(fileName))
            uploadId = multipartUpload.uploadId
            partNumber = i + 1 // Part numbers begin at 1.
            checksumAlgorithm = ChecksumAlgorithm.Sha1
        }

        CompletedPart {
```

```
            eTag = uploadPartResponse.eTag
            partNumber = i + 1
            checksumSha1 = uploadPartResponse.checksumSha1
        }
    }

s3.completeMultipartUpload {
    uploadId = multipartUpload.uploadId
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    multipartUpload {
        parts = completedParts
    }
}
```

オブジェクトのダウンロード

getObject 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、ビルダーのchecksumModeプロパティGetObjectRequestが に設定されている場合、 は に設定されますChecksumMode.Enabled。

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。

```
val request = GetObjectRequest {
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    checksumMode = ChecksumMode.Enabled
}
```

> **Note**
>
> 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。

SDK バージョン 1.4.0 以降を使用している場合、SDK はgetObjectリクエストchecksumMode = ChecksumMode.Enabledに を追加せずにリクエストの整合性を自動的にチェックします。

非同期検証

SDK for Kotlin は Amazon S3 からオブジェクトをダウンロードするときにストリーミングレスポンスを使用するため、チェックサムはオブジェクトを使用するときに計算されます。そのため、チェックサムが検証されるようにオブジェクトを使用する必要があります。

次の例では、レスポンス全体を使用してチェックサムを検証する方法を示しています。

```
val request = GetObjectRequest {
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    checksumMode = checksumMode.Enabled
}

val response = s3Client.getObject(request) {
    println(response.body?.decodeToString()) // Fully consume the object.
    // The checksum is valid.
}
```

一方、次の例のコードではオブジェクトを一切使用していないため、チェックサムは検証されません。

```
s3Client.getObject(request) {
    println("Got the object.")
}
```

SDK によって計算されたチェックサムが、レスポンスで送信される予定のチェックサムと一致しない場合、SDK は ChecksumMismatchException を投げます。

SDK for Kotlin を使用して Amazon S3 マルチリージョンアクセスポイントを操作する

Amazon S3 マルチリージョンアクセスポイントを使用すると、アプリケーションが複数の AWS リージョンにあるAmazon S3 バケットからのリクエストを実行するために使用できるグローバルエンドポイントを作成できます。マルチリージョンアクセスポイントを使用して、単一のリージョンで使用されるのと同じアーキテクチャでマルチリージョンアプリケーションを構築し、世界中のどこでもこれらのアプリケーションを実行することができます。

Amazon S3ユーザーガイド」には、マルチリージョンアクセスポイントに関する詳細な背景情報が含まれています。

マルチリージョンアクセスポイントの使用

マルチリージョンアクセスポイントを作成するには、まず、リクエストを処理する AWS リージョンごとに 1 つのバケットを指定します。次のスニペットでは、2 つのバケットを作成します。

バケットを作成する

次の 関数は、マルチリージョンアクセスポイントを操作する 2 つのバケットを作成します。1 つのバケットはリージョン にありus-east-1、もう 1 つのバケットはリージョン にありますus-west-1。

最初の引数として渡された S3 クライアントの作成は、最初の例の の下に表示されますthe section called “ オブジェクトの操作”。

```
        suspend fun setUpTwoBuckets(
            s3: S3Client,
            bucketName1: String,
            bucketName2: String,
        ) {
            println("Create two buckets in different regions.")
            // The shared aws config file configures the default Region to be us-
east-1.
            s3.createBucket(
                CreateBucketRequest {
                    bucket = bucketName1
                },
            )
            s3.waitUntilBucketExists {
                bucket = bucketName1
            }
            println("  Bucket [$bucketName1] created.")

            // Override the S3Client to work with us-west-1 for the second bucket.
            s3.withConfig {
                region = "us-west-1"
            }.use { s3West ->
                s3West.createBucket(
                    CreateBucketRequest {
                        bucket = bucketName2
                        createBucketConfiguration = CreateBucketConfiguration {
                            locationConstraint = BucketLocationConstraint.UsWest1
                        }
                    },
```

```
                )
                s3West.waitUntilBucketExists {
                    bucket = bucketName2
                }
                println("  Bucket [$bucketName2] created.")
            }
        }
```

Kotlin SDK の S3 コントロールクライアントを使用して、マルチリージョンアクセスポイントに関する情報を作成、削除、取得します。

次のスニペットに示すように、S3 コントロールアーティファクトへの依存関係を追加します。(*X.Y.Z* リンクに移動して、利用可能な最新バージョンを確認できます)。

```
...
implementation(platform("aws.sdk.kotlin:bom:X.Y.Z"))
implementation("aws.sdk.kotlin:s3control")
...
```

次のコードに示すように、 を使用する AWS リージョン us-west-2ように S3 コントロールクライアントを設定します。すべての S3 コントロールクライアントオペレーションは、 us-west-2リージョンをターゲットにする必要があります。

```
        suspend fun createS3ControlClient(): S3ControlClient {
            // Configure your S3ControlClient to send requests to US West (Oregon).
            val s3Control = S3ControlClient.fromEnvironment {
                region = "us-west-2"
            }
            return s3Control
        }
```

S3 コントロールクライアントを使用して、次のコードに示すようにバケット名 (以前に作成したもの) を指定してマルチリージョンアクセスポイントを作成します。

```
    suspend fun createMrap(
        s3Control: S3ControlClient,
        accountIdParam: String,
        bucketName1: String,
        bucketName2: String,
        mrapName: String,
```

```
    ): String {
        println("Creating MRAP ...")
        val createMrapResponse: CreateMultiRegionAccessPointResponse =
            s3Control.createMultiRegionAccessPoint {
                accountId = accountIdParam
                clientToken = UUID.randomUUID().toString()
                details {
                    name = mrapName
                    regions = listOf(
                        Region {
                            bucket = bucketName1
                        },
                        Region {
                            bucket = bucketName2
                        },
                    )
                }
            }
        val requestToken: String? = createMrapResponse.requestTokenArn

        // Use the request token to check for the status of the
 CreateMultiRegionAccessPoint operation.
        if (requestToken != null) {
            waitForSucceededStatus(s3Control, requestToken, accountIdParam)
            println("MRAP created")
        }

        val getMrapResponse =
            s3Control.getMultiRegionAccessPoint(
                input = GetMultiRegionAccessPointRequest {
                    accountId = accountIdParam
                    name = mrapName
                },
            )
        val mrapAlias = getMrapResponse.accessPoint?.alias
        return "arn:aws:s3::$accountIdParam:accesspoint/$mrapAlias"
    }
```

マルチリージョンアクセスポイントの作成は非同期オペレーションであるため、即時の応答から受け取ったトークンを使用して作成プロセスのステータスを確認します。ステータスチェックが成功メッセージを返した後、 GetMultiRegionAccessPointオペレーションを使用してマルチリージョンアクセスポイントのエイリアスを取得できます。エイリアスは、オブジェクトレベルのオペレーションに必要な ARN の最後のコンポーネントです。

トークンを使用してステータスを確認する

を使用して、最後のオペレーションのステータスDescribeMultiRegionAccessPointOperationを確認します。requestStatus 値が「成功」になったら、マルチリージョンアクセスポイントを使用できます。

```
        suspend fun waitForSucceededStatus(
            s3Control: S3ControlClient,
            requestToken: String,
            accountIdParam: String,
            timeBetweenChecks: Duration = 1.minutes,
        ) {
            var describeResponse: DescribeMultiRegionAccessPointOperationResponse
            describeResponse = s3Control.describeMultiRegionAccessPointOperation(
                input = DescribeMultiRegionAccessPointOperationRequest {
                    accountId = accountIdParam
                    requestTokenArn = requestToken
                },
            )

            var status: String? = describeResponse.asyncOperation?.requestStatus
            while (status != "SUCCEEDED") {
                delay(timeBetweenChecks)
                describeResponse = s3Control.describeMultiRegionAccessPointOperation(
                    input = DescribeMultiRegionAccessPointOperationRequest {
                        accountId = accountIdParam
                        requestTokenArn = requestToken
                    },
                )
                status = describeResponse.asyncOperation?.requestStatus
                println(status)
            }
        }
```

オブジェクトとマルチリージョンアクセスポイントの操作

S3 クライアントを使用して、マルチリージョンアクセスポイントのオブジェクトを操作します。マルチリージョンアクセスポイントで使用できるバケット内のオブジェクトで使用するオペレーションの多くは、次のとおりです。オペレーションの詳細と詳細なリストについては、「マルチリージョンアクセスポイントと S3 オペレーションとの互換性」を参照してください。

マルチリージョンアクセスポイントを使用したオペレーションは、非対称 SigV4 (SigV4a) 署名アルゴリズムで署名されます。SigV4a を設定するには、まず次の依存関係をプロジェクトに追加します。(*X.Y.Z* リンクに移動して、利用可能な最新バージョンを確認できます)。

```
...
implementation(platform("aws.sdk.kotlin:bom:X.Y.Z"))
implementation(platform("aws.smithy.kotlin:bom:X.Y.Z"))

implementation("aws.smithy.kotlin:aws-signing-default")
implementation("aws.smithy.kotlin:http-auth-aws")
implementation("aws.sdk.kotlin:s3")
...
```

依存関係を追加したら、次のコードに示すように、SigV4a 署名アルゴリズムを使用するように S3 クライアントを設定します。

```
        suspend fun createS3Client(): S3Client {
            // Configure your S3Client to use the Asymmetric SigV4 (SigV4a) signing
algorithm.
            val sigV4aScheme = SigV4AsymmetricAuthScheme(DefaultAwsSigner)
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4aScheme)
            }
            return s3
        }
```

S3 クライアントを設定すると、S3 がマルチリージョンアクセスポイントに対してサポートするオペレーションも同じように機能します。唯一の違いは、バケットパラメータがマルチリージョンアクセスポイントの ARN であることです。ARN はAmazon S3コンソールから取得することも、ARN を返すcreateMrap関数で前述したようにプログラムで取得することもできます。

次のコード例は、 GetObjectオペレーションで使用される ARN を示しています。

```
    suspend fun getObjectFromMrap(
        s3: S3Client,
        mrapArn: String,
        keyName: String,
    ): String? {
        val request = GetObjectRequest {
            bucket = mrapArn // Use the ARN instead of the bucket name for object
operations.
```

```
            key = keyName
        }

        var stringObj: String? = null
        s3.getObject(request) { resp ->
            stringObj = resp.body?.decodeToString()
            if (stringObj != null) {
                println("Successfully read $keyName from $mrapArn")
            }
        }
        return stringObj
    }
```

を使用して DynamoDB を操作する AWS SDK for Kotlin

AWS アカウントベースのエンドポイントを使用する

DynamoDB はAWS 、アカウント ID を使用してリクエストのルーティングを合理化することで、パフォーマンスを向上させることができるアカウントベースのエンドポイントを提供します。 AWS

この機能を利用するには、 のバージョン 1.3.37 以降を使用する必要があります AWS SDK for Kotlin。最新バージョンの SDK は、Maven 中央リポジトリに記載されています。サポートされているバージョンの SDK がアクティブになると、新しいエンドポイントが自動的に使用されます。

アカウントベースのルーティングをオプトアウトする場合は、次の 4 つのオプションがあります。

- を AccountIdEndpointModeに設定して DynamoDB サービスクライアントを設定しますDISABLED。
- 環境変数を設定します。
- JVM システムプロパティを設定します。
- 共有 AWS config ファイル設定を更新します。

次のスニペットは、DynamoDB サービスクライアントを設定してアカウントベースのルーティングを無効にする方法の例です。

```
DynamoDbClient.fromEnvironment {
    accountIdEndpointMode = AccountIdEndpointMode.DISABLED // The default value is
 PREFERRED.
```

```
}
```

AWS SDKs とツールのリファレンスガイド」には、最後の 3 つの設定オプションに関する詳細が記載されています。

DynamoDB マッパーを使用してクラスを DynamoDB 項目にマッピングする (開発者プレビュー)

⚠ DynamoDB Mapper はデベロッパープレビューリリースです。機能は完全ではなく、変更される可能性があります。

DynamoDB Mapper は、Kotlin クラスを DynamoDB テーブルとインデックスにマッピングするメカニズムを提供する高レベルライブラリです。DynamoDB AWS SDK for Java拡張クライアントや オブジェクト永続性モデルに似 AWS SDK for .NETています。 DynamoDB

データオブジェクトを記述するスキーマと、それらを DynamoDB 項目に変換する方法を定義します。スキーマを定義すると、DynamoDB Mapper は、テーブルとインデックスの作成、読み取り、更新、削除 (CRUD) オペレーションでオブジェクトを使用するための直感的なインターフェイスを提供します。

トピック

- DynamoDB Mapper の使用を開始する
- DynamoDB マッパーを設定する
- 注釈からスキーマを生成する
- スキーマを手動で定義する
- DynamoDB Mapper でセカンダリインデックスを使用する
- 式を使用する

DynamoDB Mapper の使用を開始する

⚠ DynamoDB Mapper はデベロッパープレビューリリースです。機能は完了しておらず、変更される可能性があります。

次のチュートリアルでは、DynamoDB Mapper の基本的なコンポーネントを紹介し、コードで使用する方法を示します。

依存関係を追加する

Gradle プロジェクトで DynamoDB Mapper の使用を開始するには、プラグインと 2 つの依存関係を `build.gradle.kts` ファイルに追加します。

(*X.Y.Z* リンクに移動して、利用可能な最新バージョンを確認できます)。

```
// build.gradle.kts
val sdkVersion: String = X.Y.Z

plugins {
    id("aws.sdk.kotlin.hll.dynamodbmapper.schema.generator") version "$sdkVersion-
beta" // For the Developer Preview, use the beta version of the latest SDK.
}

dependencies {
    implementation("aws.sdk.kotlin:dynamodb-mapper:$sdkVersion-beta")
    implementation("aws.sdk.kotlin:dynamodb-mapper-annotations:$sdkVersion-beta")
}
```

**<Version>* を SDK の最新リリースに置き換えます。SDK の最新バージョンを確認するには、GitHub で最新リリースを確認してください。

> **Note**
>
> スキーマを手動で定義する場合は、これらの依存関係の一部はオプションです。依存関係の詳細と削減されたセットthe section called “スキーマを手動で定義する”については、「」を参照してください。

マッパーを作成して使用する

DynamoDB Mapper は、 AWS SDK for Kotlin DynamoDB クライアントを使用して DynamoDB とやり取りします。次のコードスニペットに示すように、マッパーDynamoDbClientインスタンスを作成するときに、完全に設定されたインスタンスを指定する必要があります。

```
import aws.sdk.kotlin.hll.dynamodbmapper.DynamoDbMapper
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient
```

```
val client = DynamoDbClient.fromEnvironment()
val mapper = DynamoDbMapper(client)
```

マッパーインスタンスを作成したら、次に示すようにそれを使用してテーブルインスタンスを取得できます。

```
val carsTable = mapper.getTable("cars", CarSchema)
```

前のコードでは、 で定義されたスキーマcarsを使用して、 というDynamoDB名前の テーブルへの参照を取得します CarSchema (以下でスキーマについて説明します) 。テーブルインスタンスを作成したら、そのインスタンスに対してオペレーションを実行できます。次のコードス二ペットは、carsテーブルに対する 2 つのオペレーションの例を示しています。

```
carsTable.putItem {
    item = Car(make = "Ford", model = "Model T", ...)
}

carsTable
   .queryPaginated {
        keyCondition = KeyFilter(partitionKey = "Peugeot")
   }
   .items()
   .collect { car -> println(car) }
```

前のコードでは、carsテーブルに新しい項目が作成されます。このコードでは、 Car クラスを使用してCarインスタンスをインラインで作成します。定義は以下のとおりです。次に、コードはパーティションキーが である項目をcarsテーブルにクエリPeugeotし、それらを出力します。オペレーションの詳細については、以下を参照してください。

クラス注釈を使用してスキーマを定義する

さまざまな Kotlin クラスの場合、SDK は Gradle 用の DynamoDB Mapper Schema Generator プラグインを使用して、ビルド時にスキーマを自動的に生成できます。スキーマジェネレーターを使用すると、SDK はクラスを検査してスキーマを推測します。これにより、スキーマを手動で定義する際の共通部分の一部が軽減されます。追加の注釈と設定を使用して、生成されるスキーマをカスタマイズできます。

注釈からスキーマを生成するには、まず、クラスに @DynamoDbItemと、@DynamoDbPartitionKeyおよび のキーに注釈を付けます@DynamoDbSortKey。次のコードは、注釈付きCarクラスを示しています。

```
// The annotations used in the Car class are used by the plugin to generate a schema.
@DynamoDbItem
data class Car(
    @DynamoDbPartitionKey
    val make: String,

    @DynamoDbSortKey
    val model: String,

    val initialYear: Int
)
```

構築後、自動的に生成された を参照できますCarSchema。以下に示すように、マッパーの getTableメソッドで リファレンスを使用してテーブルインスタンスを取得できます。

```
import aws.sdk.kotlin.hll.dynamodbmapper.generatedschemas.CarSchema

// `CarSchema` is generated at build time.
val carsTable = mapper.getTable("cars", CarSchema)
```

または、ビルド時にDynamoDbMapper自動的に生成される の拡張機能メソッドを利用して、テーブルインスタンスを取得することもできます。このアプローチを使用すると、スキーマを名前で参照する必要はありません。以下に示すように、自動生成されたgetCarsTable拡張メソッドはテーブルインスタンスへの参照を返します。

```
val carsTable = mapper.getCarsTable("cars")
```

詳細と例については、「the section called “スキーマを生成する”」を参照してください。

呼び出しオペレーション

DynamoDB Mapper は、SDK の で使用できるオペレーションのサブセットをサポートしますDynamoDbClient。マッパーオペレーションの名前は、SDK クライアントの対応するオペレーションと同じです。マッパーのリクエスト/レスポンスメンバーの多くは SDK クライアントのメンバーと同じですが、名前変更、再入力、削除されたメンバーもあります。

次に示すように、DSL 構文を使用してテーブルインスタンスで オペレーションを呼び出します。

```
import aws.sdk.kotlin.hll.dynamodbmapper.operations.putItem
import aws.sdk.kotlin.services.dynamodb.model.ReturnConsumedCapacity
```

```
val putResponse = carsTable.putItem {
    item = Car(make = "Ford", model = "Model T", ...)
    returnConsumedCapacity = ReturnConsumedCapacity.Total
}

println(putResponse.consumedCapacity)
```

明示的なリクエストオブジェクトを使用してオペレーションを呼び出すこともできます。

```
import aws.sdk.kotlin.hll.dynamodbmapper.operations.PutItemRequest
import aws.sdk.kotlin.services.dynamodb.model.ReturnConsumedCapacity

val putRequest = PutItemRequest<Car> {
    item = Car(make = "Ford", model = "Model T", ...)
    returnConsumedCapacity = ReturnConsumedCapacity.Total
}

val putResponse = carsTable.putItem(putRequest)
println(putResponse.consumedCapacity)
```

前の 2 つのコード例は同等です。

ページ分割されたレスポンスを使用する

query や などの一部のオペレーションでは、大きすぎて 1 回のレスポンスで返せない可能性があるデータ収集を返すscanことができます。すべてのオブジェクトが処理されるようにするために、DynamoDB Mapper はページ分割メソッドを提供します。このメソッドは DynamoDB Flowをすぐに呼び出すのではなく、次Flow<ScanResponse<Car>>に示すように、オペレーションレスポンスタイプの を返します。

```
import aws.sdk.kotlin.hll.dynamodbmapper.operations.scanPaginated

val scanResponseFlow = carsTable.scanPaginated { }

scanResponseFlow.collect { response ->
    val items = response.items.orEmpty()
    println("Found page with ${items.size} items:")

    items.forEach { car -> println(car) }
}
```

多くの場合、オブジェクトのフローは、オブジェクトを含むレスポンスのフローよりもビジネスロジックに役立ちます。マッパーは、オブジェクトのフローにアクセスするためのページ分割されたレスポンスの拡張メソッドを提供します。例えば、次のコードは、前に示した `Flow<ScanResponse<Car>>` `Flow<Car>`ではなく を返します。

```
import aws.sdk.kotlin.hll.dynamodbmapper.operations.items
import aws.sdk.kotlin.hll.dynamodbmapper.operations.scanPaginated

val carFlow = carsTable
    .scanPaginated { }
    .items()

carFlow.collect { car -> println(car) }
```

DynamoDB マッパーを設定する

> ⚠ DynamoDB Mapper はデベロッパープレビューリリースです。機能は完全ではなく、変更される可能性があります。

DynamoDB Mapper には、アプリケーションに合わせてライブラリの動作をカスタマイズできる設定オプションが用意されています。

インターセプターを使用する

DynamoDB マッパーライブラリは、マッパーのリクエストパイプラインの重要な段階で利用できるフックを定義します。`Interceptor` インターフェイスを実装して、マッパープロセスを監視または変更するフックを実装できます。

1 つの DynamoDB Mapper に設定オプションとして 1 つ以上のインターセプターを登録できます。インターセプターを登録する方法については、このセクションの最後にある例を参照してください。

リクエストパイプラインを理解する

マッパーのリクエストパイプラインは、次の 5 つのステップで構成されます。

1. 初期化： オペレーションを設定し、初期コンテキストを収集します。
2. シリアル化： 高レベルのリクエストオブジェクトを低レベルのリクエストオブジェクトに変換します。このステップでは、高レベルの Kotlin オブジェクトを属性名と値で構成される DynamoDB 項目に変換します。

3. 低レベル呼び出し： 基盤となる DynamoDB クライアントでリクエストを実行します。
4. 逆シリアル化： 低レベルのレスポンスオブジェクトを高レベルのレスポンスオブジェクトに変換します。このステップでは、属性名と値で構成される DynamoDB 項目を高レベルの Kotlin オブジェクトに変換します。
5. 完了： 発信者に返すために高レベルのレスポンスを確定します。パイプラインの実行中に例外がスローされた場合、このステップは呼び出し元にスローされる例外を確定します。

フック

フックは、マッパーがパイプラインの特定のステップの前後に呼び出すインターセプターメソッドです。フックには、読み取り専用と変更 (または読み取り/書き込み) の 2 つのバリアントがあります。例えば、 `readBeforeInvocation`は、低レベルの呼び出しステップの前に フェーズでマッパーが実行する読み取り専用フックです。

読み取り専用フック

マッパーは、パイプラインの各ステップの前後に読み取り専用フックを呼び出します (初期化ステップの前と後にを除く)。読み取り専用のフードは、進行中の高レベルのオペレーションの読み取り専用ビューを提供します。ログ記録、デバッグ、メトリクスの収集などのオペレーションの状態を調べるメカニズムを提供します。各読み取り専用フックはコンテキスト引数を受け取り、 を返しますUnit。

マッパーは、読み取り専用フック中にスローされた例外をキャッチし、コンテキストに追加します。次に、同じフェーズの後続のインターセプターフックに例外でコンテキストを渡します。マッパーは、同じフェーズで最後のインターセプターの読み取り専用フックを呼び出した後にのみ、呼び出し元に例外をスローします。例えば、マッパーに 2 つのインターセプター Aと が設定されBていて、 AのreadAfterSerializationフックが例外をスローした場合、マッパーは BのreadAfterSerializationフックに渡されたコンテキストに例外を追加します。BのreadAfterSerializationフックが完了すると、マッパーは例外を呼び出し元にスローします。

フックの変更

マッパーは、パイプラインの各ステップの前に変更フックを呼び出します (初期化前を除く)。変更フックは、進行中の高レベルのオペレーションを表示および変更する機能を提供します。マッパー設定や項目スキーマではカスタマイズできない方法で動作やデータをカスタマイズするために使用できます。各変更フックはコンテキスト引数を受け取り、その結果としてそのコンテキストの一部のサブセットを返します。これは、フックによって変更されるか、入力コンテキストから渡されます。

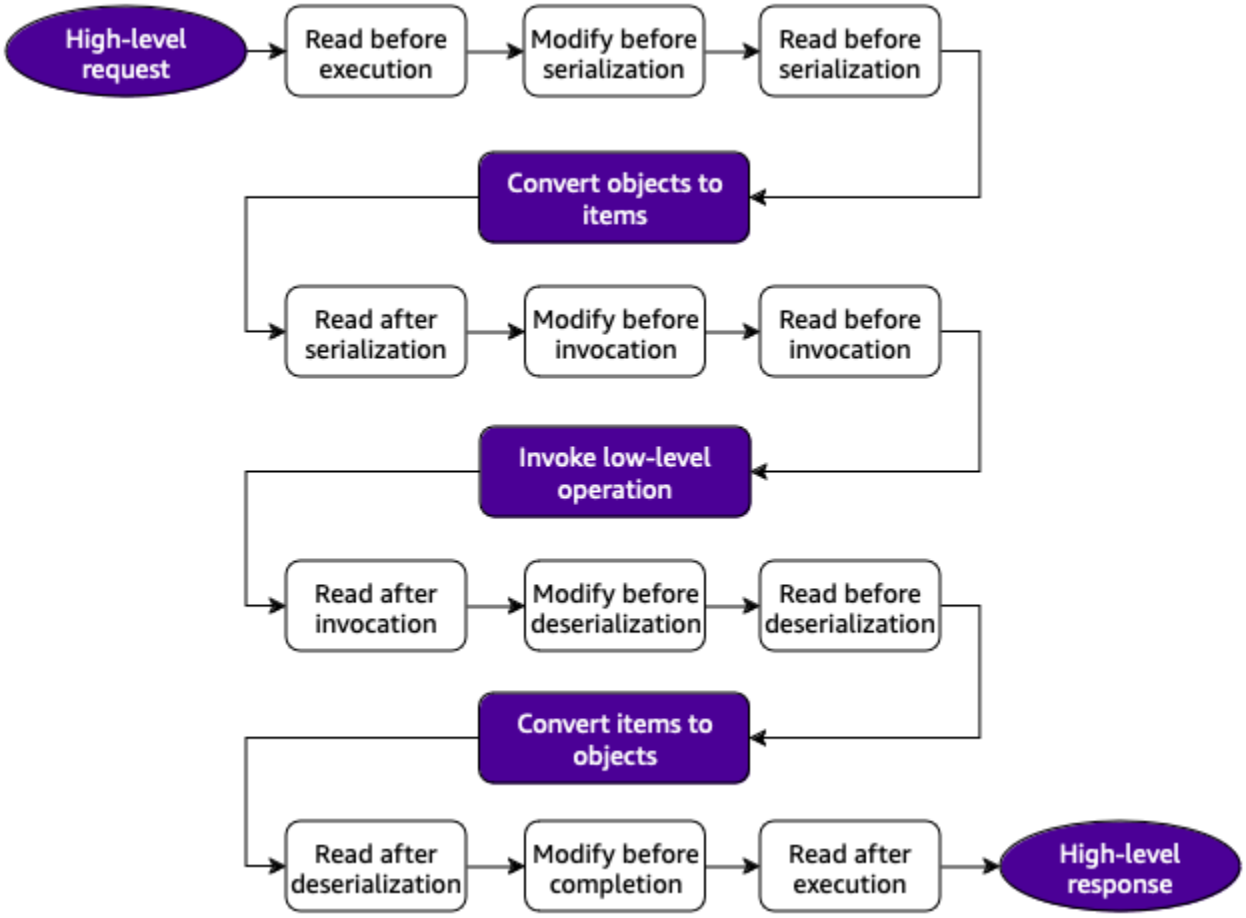
変更フックの実行中にマッパーが例外をキャッチした場合、他のインターセプターの変更フックは同じフェーズで実行されません。マッパーは例外をコンテキストに追加し、次の読み取り専用フックに渡します。マッパーは、同じフェーズで最後のインターセプターの読み取り専用フックを呼び出した後にのみ、呼び出し元に例外をスローします。例えば、マッパーが 2 つのインターセプター Aと で構成されていてB、 AのmodifyBeforeSerializationフックが例外をスローした場合、 BのmodifyBeforeSerializationフックは呼び出されません。インターセプタAーの およびB'readAfterSerializationフックが実行され、その後例外が呼び出し元にスローバックされます。

実行順序

マッパーの設定でインターセプターが定義される順序によって、マッパーがフックを呼び出す順序が決まります。

- 低レベル呼び出しステップの前にあるフェーズでは、設定で追加されたのと同じ順序でフックを実行します。
- 低レベル呼び出しステップの後のフェーズでは、設定に追加された順序と逆の順序でフックを実行します。

次の図は、フックメソッドの実行順序を示しています。



フックメソッドの実行順序のテキスト説明

マッパーはインターセプターのフックを次の順序で実行します。

1. DynamoDB Mapper が高レベルのリクエストを呼び出します
2. 実行前に読み取る
3. シリアル化前に変更する
4. シリアル化前に読み取る
5. DynamoDB Mapper はオブジェクトを項目に変換します
6. シリアル化後の読み取り
7. 呼び出し前に変更する
8. 呼び出し前に読み取る
9. DynamoDB Mapper が低レベルオペレーションを呼び出します
10. 呼び出し後の読み取り
11. 逆シリアル化前に変更する

12逆シリアル化前に読み取る

13DynamoDB Mapper は項目をオブジェクトに変換します

14逆シリアル化後の読み取り

15完了前に を変更する

16実行後の読み取り

17DynamoDB Mapper が高レベルのレスポンスを返す

設定例

次の例は、DynamoDbMapperインスタンスでインターセプターを設定する方法を示しています。

```
import aws.sdk.kotlin.hll.dynamodbmapper.DynamoDbMapper
import aws.sdk.kotlin.hll.dynamodbmapper.operations.ScanRequest
import aws.sdk.kotlin.hll.dynamodbmapper.operations.ScanResponse
import aws.sdk.kotlin.hll.dynamodbmapper.pipeline.Interceptor
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient
import aws.sdk.kotlin.services.dynamodb.model.ScanRequest as LowLevelScanRequest
import aws.sdk.kotlin.services.dynamodb.model.ScanResponse as LowLevelScanResponse

val printingInterceptor = object : Interceptor<User, ScanRequest<User>,
 LowLevelScanRequest, LowLevelScanResponse, ScanResponse<User>> {
    override fun readBeforeDeserialization(ctx: LResContext<User, ScanRequest<User>,
 LowLevelScanRequest, LowLevelScanResponse>) {
        println("Scan response contains ${ctx.lowLevelResponse.count} items.")
    }
}

val client = DynamoDbClient.fromEnvironment()

val mapper = DynamoDbMapper(client) {
    interceptors += printingInterceptor
}
```

注釈からスキーマを生成する

⚠ DynamoDB Mapper はデベロッパープレビューリリースです。機能は完全ではなく、変更される可能性があります。

DynamoDB Mapper は、Kotlin クラスと DynamoDB 項目間のマッピングを定義するスキーマに依存します。Kotlin クラスは、スキーマジェネレータ Gradle プラグインを使用してスキーマの作成を駆動できます。

プラグインを適用する

クラスのコード生成スキーマを開始するには、アプリケーションのビルドスクリプトに プラグインを適用し、注釈モジュールへの依存関係を追加します。次の Gradle スクリプトスニペットは、コード生成に必要なセットアップを示しています。

(*X.Y.Z* リンクに移動して、利用可能な最新バージョンを確認できます)。

```
// build.gradle.kts
val sdkVersion: String = X.Y.Z

plugins {
    id("aws.sdk.kotlin.hll.dynamodbmapper.schema.generator") version "$sdkVersion-
beta" // For the Developer Preview, use the beta version of the latest SDK.
}

dependencies {
    implementation("aws.sdk.kotlin:dynamodb-mapper:$sdkVersion-beta")
    implementation("aws.sdk.kotlin:dynamodb-mapper-annotations:$sdkVersion-beta")
}
```

プラグインを設定する

プラグインには、ビルドスクリプトでdynamoDbMapper { ... }プラグイン拡張機能を使用して適用できるいくつかの設定オプションが用意されています。

オプション	オプションの説明	値
generateBuilderClasses	注釈が付けられたクラスに対して DSL スタイルのビルダークラスを生成するかどうかを制御します @DynamoDbItem	WHEN_REQUIRED (デフォルト): ビルダークラスは、パブリック可変メンバーのみで構成され、ゼロ引数コンストラクタを持つクラスでは生成されません ALWAYS: ビルダークラスは常に生成されます

オプション	オプションの説明	値
visibility	生成されたクラスの可視性を制御します	PUBLIC (デフォルト) INTERNAL
destinationPackage	生成されたクラスのパッケージ名を指定します。	RELATIVE (デフォルト): スキーマクラスは、注釈付きクラスを基準にしたサブパッケージで生成されます。デフォルトでは、サブパッケージには dynamodbmapper.generatedschemas という名前が付けられ、文字列パラメータを渡すことで設定できます。 ABSOLUTE: スキーマクラスは、アプリケーションのルートを基準にした絶対パッケージで生成されます。デフォルトでは、パッケージの名前はでaws.sdk.kotlin.hll.dynamodbmapper.generatedschemas 、これは文字列パラメータを渡すことで設定できます。
generateGetTableExtension	DynamoDbMapper.get${CLASS_NAME}Table 拡張メソッドを生成するかどうかを制御します	true (デフォルト) false

Example コード生成プラグイン設定の例

次の例では、生成されたスキーマの送信先パッケージと可視性を設定します。

```
// build.gradle.kts
```

```
import aws.sdk.kotlin.hll.dynamodbmapper.codegen.annotations.DestinationPackage
import aws.sdk.kotlin.hll.dynamodbmapper.codegen.annotations.Visibility
import aws.smithy.kotlin.runtime.ExperimentalApi

@OptIn(ExperimentalApi::class)
dynamoDbMapper {
    destinationPackage = DestinationPackage.RELATIVE("my.configured.package")
    visibility = Visibility.INTERNAL
}
```

クラスに注釈を付ける

スキーマジェネレーターは、クラス注釈を検索して、スキーマを生成するクラスを決定します。スキーマの生成をオプトインするには、 でクラスに注釈を付けます@DynamoDbItem。また、 注釈を使用して、項目のパーティションキーとして機能するクラスプロパティに@DynamoDbPartitionKey注釈を付ける必要があります。

次のクラス定義は、スキーマ生成に必要な最小限の注釈を示しています。

Example

```
@DynamoDbItem
data class Employee(
    @DynamoDbPartitionKey
    val id: Int,

    val name: String,
    val role: String,
)
```

クラス注釈

スキーマの生成を制御するために、次の注釈がクラスに適用されます。

- @DynamoDbItem: このクラス/インターフェイスがテーブル内の項目タイプを記述することを指定します。このタイプのすべてのパブリックプロパティは、明示的に無視されない限り、属性にマッピングされます。存在する場合、このクラスのスキーマが生成されます。
  - converterName: スキーマジェネレータプラグインによって作成されたスキーマではなく、カスタムスキーマを使用する必要があることを示すオプションのパラメータ。これはカスタムItemConverterクラスの完全修飾名です。このセクションでは、カスタムスキーマを作成して使用する例the section called “カスタム項目コンバーターを定義する”を示します。

プロパティ注釈

次の注釈をクラスプロパティに適用して、スキーマの生成を制御できます。

- @DynamoDbPartitionKey: 項目のパーティションキーを指定します。
- @DynamoDbSortKey: 項目のオプションのソートキーを指定します。
- @DynamoDbIgnore: このクラスプロパティを DynamoDB Mapper によって Item 属性との間で変換しないように指定します。
- @DynamoDbAttribute: このクラスプロパティのオプションカスタム属性名を指定します。

カスタム項目コンバーターを定義する

場合によっては、 クラスのカスタム項目コンバーターを定義できます。その理由の 1 つは、クラスがスキーマジェネレータプラグインでサポートされていないタイプを使用している場合です。例として、次のバージョンの Employee クラスを使用します。

```
import kotlin.uuid.Uuid

@DynamoDbItem
data class Employee(
    @DynamoDbPartitionKey
    var id: Int,

    var name: String,
    var role: String,
    var workstationId: Uuid
)
```

Employee クラスは kotlin.uuid.Uuidタイプを使用するようになりました。これはスキーマジェネレーターで現在サポートされていません。スキーマの生成が失敗し、エラー が発生しますUnsupported attribute type TypeRef(pkg=kotlin.uuid, shortName=Uuid, genericArgs=[], nullable=false)。このエラーは、プラグインがこのクラスの項目コンバータを生成できないことを示します。したがって、独自の を記述する必要があります。

これを行うには、 ItemConverter クラスの を実装し、新しい項目コンバータの完全修飾名を指定して@DynamoDbItemクラス注釈を変更します。

まず、 kotlin.uuid.Uuid クラスValueConverterに を実装します。

```
import aws.sdk.kotlin.hll.dynamodbmapper.values.ValueConverter
```

```
import aws.sdk.kotlin.services.dynamodb.model.AttributeValue
import kotlin.uuid.Uuid

public val UuidValueConverter = object : ValueConverter<Uuid> {
    override fun convertFrom(to: AttributeValue): Uuid =
        Uuid.parseHex(to.asS())

    override fun convertTo(from: Uuid): AttributeValue =
        AttributeValue.S(from.toHexString())
}
```

次に、 Employee クラスItemConverterに を実装します。は、「workstationId」の属性記述子でこの新しい値コンバータItemConverterを使用します。

```
import aws.sdk.kotlin.hll.dynamodbmapper.items.AttributeDescriptor
import aws.sdk.kotlin.hll.dynamodbmapper.items.ItemConverter
import aws.sdk.kotlin.hll.dynamodbmapper.items.SimpleItemConverter
import aws.sdk.kotlin.hll.dynamodbmapper.values.scalars.IntConverter
import aws.sdk.kotlin.hll.dynamodbmapper.values.scalars.StringConverter

public object MyEmployeeConverter : ItemConverter<Employee> by SimpleItemConverter(
    builderFactory = { Employee() },
    build = { this },
    descriptors = arrayOf(
        AttributeDescriptor(
            "id",
            Employee::id,
            Employee::id::set,
            IntConverter,
        ),
        AttributeDescriptor(
            "name",
            Employee::name,
            Employee::name::set,
            StringConverter,
        ),
        AttributeDescriptor(
            "role",
            Employee::role,
            Employee::role::set,
            StringConverter
        ),
        AttributeDescriptor(
```

```
            "workstationId",
            Employee::workstationId,
            Employee::workstationId::set,
            UuidValueConverter
        )
    ),
)
```

項目コンバーターを定義したので、 クラスに適用できます。以下に示すように、完全修飾クラス名を指定して項目コンバーターを参照するように@DynamoDbItem注釈を更新します。

```
import kotlin.uuid.Uuid

@DynamoDbItem("my.custom.item.converter.MyEmployeeConverter")
data class Employee(
    @DynamoDbPartitionKey
    var id: Int,

    var name: String,
    var role: String,
    var workstationId: Uuid
)
```

最後に、DynamoDB Mapper で クラスの使用を開始できます。

スキーマを手動で定義する

⚠ DynamoDB Mapper はデベロッパープレビューリリースです。機能は完了しておらず、変更される可能性があります。

コードでスキーマを定義する

最大限の制御とカスタマイズ性を実現するために、コードでスキーマを手動で定義およびカスタマイズできます。

次のスニペットに示すように、注釈駆動型スキーマ作成を使用する場合と比較して、build.gradle.ktsファイルに含める依存関係の数を減らす必要があります。

(*X.Y.Z* リンクに移動して、利用可能な最新バージョンを確認できます)。

```
// build.gradle.kts
val sdkVersion: String = X.Y.Z

dependencies {
    implementation("aws.sdk.kotlin:dynamodb-mapper:$sdkVersion-beta") // For the
 Developer Preview, use the beta version of the latest SDK.
}
```

スキーマジェネレータープラグインや注釈パッケージは必要ありません。

Kotlin クラスと DynamoDB 項目間のマッピングにはItemSchema<T>実装が必要です。ここで、Tは Kotlin クラスのタイプです。スキーマは以下の要素で構成されます。

- Kotlin オブジェクトインスタンスと DynamoDB 項目間の変換方法を定義する項目コンバータ。
- パーティションキー属性の名前とタイプを定義するパーティションキー仕様。
- オプションで、ソートキー属性の名前とタイプを定義するソートキー仕様。

次のコードでは、CarSchemaインスタンスを手動で作成します。

```
import aws.sdk.kotlin.hll.dynamodbmapper.items.ItemConverter
import aws.sdk.kotlin.hll.dynamodbmapper.items.ItemSchema
import aws.sdk.kotlin.hll.dynamodbmapper.model.itemOf

// We define a schema for this data class.
data class Car(val make: String, val model: String, val initialYear: Int)

// First, define an item converter.
val carConverter = object : ItemConverter<Car> {
    override fun convertTo(from: Car, onlyAttributes: Set<String>?): Item  = itemOf(
        "make" to AttributeValue.S(from.make),
        "model" to AttributeValue.S(from.model),
        "initialYear" to AttributeValue.N(from.initialYear.toString()),
    )

    override fun convertFrom(to: Item): Car = Car(
        make = to["make"]?.asSOrNull() ?: error("Invalid attribute `make`"),
        model = to["model"]?.asSOrNull() ?: error("Invalid attribute `model`"),
        initialYear = to["initialYear"]?.asNOrNull()?.toIntOrNull()
            ?: error("Invalid attribute `initialYear`"),
    )
}
```

```
// Next, define the specifications for the partition key and sort key.
val makeKey = KeySpec.String("make")
val modelKey = KeySpec.String("model")

// Finally, create the schema from the converter and key specifications.
// Note that the KeySpec for the partition key comes first in the ItemSchema
 constructor.
val CarSchema = ItemSchema(carConverter, makeKey, modelKey)
```

前のコードでは、 という名前のコンバーターを作成します。これはcarConverter、 の匿名実装として定義されますItemConverter<Car>。コンバータの convertToメソッドは引Car数を受け入れ、DynamoDB 項目属性のリテラルキーと値を表すItemインスタンスを返します。コンバータの convertFromメソッドは Item引数を受け入れ、 Item引数の属性値からCarインスタンスを返します。

次に、コードは 2 つのキー仕様を作成します。1 つはパーティションキー用、もう 1 つはソートキー用です。すべての DynamoDB テーブルまたはインデックスには、パーティションキーが 1 つだけ必要です。それに応じて、すべての DynamoDB Mapper スキーマ定義が必要です。スキーマにはソートキーが 1 つある場合もあります。

最後のステートメントでは、コードはコンバータとキーの仕様から cars DynamoDB テーブルのスキーマを作成します。

結果のスキーマは、 the section called “クラス注釈を使用してスキーマを定義する”セクションで生成した注釈駆動型スキーマと同等です。参考までに、使用した注釈付きクラスを次に示します。

DynamoDB マッパー注釈付きのカークラス

```
@DynamoDbItem
data class Car(
    @DynamoDbPartitionKey
    val make: String,

    @DynamoDbSortKey
    val model: String,

    val initialYear: Int
)
```

DynamoDB Mapper にはItemConverter、独自の の実装に加えて、次のような便利な実装がいくつか含まれています。

- SimpleItemConverter: は、ビルダークラスと属性記述子を使用してシンプルな変換ロジックを提供します。この実装を利用する方法については、「」の例the section called “カスタム項目コンバーターを定義する”を参照してください。
- HeterogeneousItemConverter: は、識別子属性とサブタイプの委任ItemConverterインスタンスを使用して多相型変換ロジックを提供します。
- DocumentConverter: Document オブジェクト内の非構造化データの変換ロジックを提供します。

DynamoDB Mapper でセカンダリインデックスを使用する

⚠ DynamoDB Mapper はデベロッパープレビューリリースです。機能は完了しておらず、変更される可能性があります。

セカンダリインデックスのスキーマを定義する

DynamoDB テーブルは、ベーステーブル自体で定義されているキーとは異なるキーを使用してデータへのアクセスを提供するセカンダリインデックスをサポートします。ベーステーブルと同様に、DynamoDB Mapper は ItemSchemaタイプを使用してインデックスとやり取りします。

DynamoDB セカンダリインデックスには、ベーステーブルのすべての属性を含める必要はありません。したがって、インデックスにマッピングされる Kotlin クラスは、そのインデックスのベーステーブルにマッピングされる Kotlin クラスとは異なる場合があります。この場合、インデックスクラスに対して別のスキーマを宣言する必要があります。

次のコードは、DynamoDB carsテーブルのインデックススキーマを手動で作成します。

```
import aws.sdk.kotlin.hll.dynamodbmapper.items.ItemConverter
import aws.sdk.kotlin.hll.dynamodbmapper.items.ItemSchema
import aws.sdk.kotlin.hll.dynamodbmapper.model.itemOf

// This is a data class for modelling the index of the Car table. Note
// that it contains a subset of the fields from the Car class and also
// uses different names for them.
data class Model(val name: String, val manufacturer: String)

// We define an item converter.
val modelConverter = object : ItemConverter<Model> {
```

```
    override fun convertTo(from: Model, onlyAttributes: Set<String>?): Item  = itemOf(
        "model" to AttributeValue.S(from.name),
        "make" to AttributeValue.S(from.manufacturer),
    )

    override fun convertFrom(to: Item): Model = Model(
        name = to["model"]?.asSOrNull() ?: error("Invalid attribute `model`"),
        manufacturer = to["make"]?.asSOrNull() ?: error("Invalid attribute `make`"),
    )
}
val modelKey = KeySpec.String("model")
val makeKey = KeySpec.String("make")

val modelSchema = ItemSchema(modelConverter, modelKey, makeKey) // The partition key
 specification is the second parameter.

/* Note that `Model` index's partition key is `model` and its sort key is `make`,
   whereas the `Car` base table uses `make` as the partition key and `model` as the
 sort key:

        @DynamoDbItem
        data class Car(
            @DynamoDbPartitionKey
            val make: String,

            @DynamoDbSortKey
            val model: String,

            val initialYear: Int
        )
*/
```

オペレーションで Model インスタンスを使用できるようになりました。

オペレーションでセカンダリインデックスを使用する

DynamoDB Mapper は、インデックスに対するオペレーションのサブセット、つまり queryPaginatedと をサポートしていますscanPaginated。インデックスでこれらのオペレーションを呼び出すには、まずテーブルオブジェクトからインデックスへの参照を取得する必要があります。次の例では、cars-by-modelインデックス用に以前にmodelSchema作成した を使用します (ここには作成は表示されません)。

```
val table = mapper.getTable("cars", CarSchema)
```

```
val index = table.getIndex("cars-by-model", modelSchema)

val modelFlow = index
    .scanPaginated { }
    .items()

modelFlow.collect { model -> println(model) }
```

式を使用する

> ⚠ DynamoDB Mapper はデベロッパープレビューリリースです。機能は完了しておらず、変更される可能性があります。

特定の DynamoDB オペレーションは、制約または条件を指定するために使用できる式を受け入れます。DynamoDB Mapper は、式を作成するためのイディオマティック Kotlin DSL を提供します。DSL を使用すると、コードの構造と読みやすさが向上し、式を簡単に記述できます。

このセクションでは、DSL 構文について説明し、さまざまな例を示します。

オペレーションで式を使用する

式は、 などのオペレーションで使用します。ここではscan、定義した条件に基づいて返された項目をフィルタリングします。DynamoDB Mapper で式を使用するには、オペレーションリクエストに式コンポーネントを追加します。

次のスニペットは、 scanオペレーションで使用されるフィルター式の例を示しています。Lambda 引数を使用して、返される項目をyear属性値が 2001 の項目に制限するフィルター条件を記述します。

```
val table = // A table instance.

table.scanPaginated {
    filter {
        attr("year") eq 2001
    }
}
```

次の例は、ソートキーフィルタリングと非キーフィルタリングの 2 つの場所で式をサポートするqueryオペレーションを示しています。

```
table.queryPaginated {
    keyCondition = KeyFilter(partitionKey = 1000) { sortKey startsWith "M" }
    filter {
        attr("year") eq 2001
    }
}
```

前のコードは、3 つの基準をすべて満たす結果にフィルタリングします。

- パーティションキー属性値は 1000-AND-
- M-AND- で始まるソートキー属性値
- year 属性値は 2001 です

DSL コンポーネント

DSL 構文は、式の構築に使用するいくつかのタイプのコンポーネントを公開します。以下に説明します。

属性

ほとんどの条件は、キーまたはドキュメントパスによって識別される属性を参照します。DSK では、 attr関数を使用してすべての属性参照を作成し、オプションで追加の変更を加えます。

次のコードは、インデックスによるリストの参照解除やキーによるマップの参照解除など、単純な属性参照から複雑な属性参照までの範囲を示しています。

```
attr("foo")           // Refers to the value of top-level attribute `foo`.

attr("foo")[3]        // Refers to the value at index 3 in the list value of
                      // attribute `foo`.

attr("foo")[3]["bar"] // Refers to the value of key `bar` in the map value at
                      // index 3 of the list value of attribute `foo`.
```

等価と不等価

式内の属性値は、等号と不等号で比較できます。属性値とリテラル値またはその他の属性値を比較できます。条件を指定するために使用する関数は次のとおりです。

- eq: が に等しい (に相当==)

- neq: が と等しくない (に相当!=)
- gt: が より大きい (に相当>)
- gte: が 以上 (に相当>=)
- lt: が より小さい (に相当<)
- lte: 以下 (に相当<=)

比較関数を引数と組み合わせるには、次の例に示すように、インフィックス表記を使用します。

```
attr("foo") eq 42           // Uses a literal. Specifies that the attribute value `foo`
 must be
                            // equal to 42.

attr("bar") gte attr("baz") // Uses another attribute value. Specifies that the
 attribute
                            // value `bar` must be greater than or equal to the
                            // attribute value of `baz`.
```

範囲とセット

単一の値に加えて、属性値を範囲またはセット内の複数の値と比較できます。次の例に示すように、infix isIn関数を使用して比較を行います。

```
attr("foo") isIn 0..99  // Specifies that the attribute value `foo` must be
                        // in the range of `0` to `99` (inclusive).

attr("foo") isIn setOf( // Specifies that the attribute value `foo` must be
    "apple",            // one of `apple`, `banana`, or `cherry`.
    "banana",
    "cherry",
)
```

このisIn関数は、コレクション (などSet<String>) と Kotlin として表現できる境界 ClosedRange<T> (など) のオーバーロードを提供しますIntRange。として表現できない境界 ClosedRange<T> (バイト配列やその他の属性参照など) には、 isBetween関数を使用できます。

```
val lowerBytes = byteArrayOf(0x48, 0x65, 0x6c)  // Specifies that the attribute value
val upperBytes = byteArrayOf(0x6c, 0x6f, 0x21)  // `foo` is between the values
attr("foo").isBetween(lowerBytes, upperBytes)   // `0x48656c` and `0x6c6f21`
```

```
attr("foo").isBetween(attr("bar"), attr("baz")) // Specifies that the attribute value
                                                // `foo` is between the values of
                                                // attributes `bar` and `baz`.
```

ブールロジック

以下の関数を使用して、個々の条件を組み合わせるか、ブールロジックを使用して変更できます。

- and: すべての条件が true (と同等 &&) である必要があります
- or: 少なくとも 1 つの条件が true (に相当 | |) である必要があります
- not: 指定された条件は false (に相当 !) である必要があります

次の例は、各関数を示しています。

```
and(                            // Both conditions must be met:
    attr("foo") eq "banana",    // * attribute value `foo` must equal `banana`
    attr("bar") isIn 0..99,     // * attribute value `bar` must be between
)                               //   0 and 99 (inclusive)

or(                             // At least one condition must be met:
    attr("foo") eq "cherry",    // * attribute value `foo` must equal `cherry`
    attr("bar") isIn 100..199, // * attribute value `bar` must be between
)                               //   100 and 199 (inclusive)

not(                            // The attribute value `foo` must *not* be
    attr("baz") isIn setOf(     // one of `apple`, `banana`, or `cherry`.
        "apple",                // Stated another way, the attribute value
        "banana",               // must be *anything except* `apple`, `banana`,
        "cherry",               // or `cherry`--including potentially a
    ),                          // non-string value or no value at all.
)
```

次の式に示すように、ブール関数でブール条件をさらに組み合わせて、ネストされたロジックを作成できます。

```
or(
    and(
        attr("foo") eq 123,
        attr("bar") eq "abc",
    ),
    and(
```

```
        attr("foo") eq 234,
        attr("bar") eq "bcd",
    ),
)
```

前の式は、次のいずれかの条件を満たすものに結果をフィルタリングします。

- これらの条件は両方とも true です。
  - foo 属性値は 123-AND-
  - bar 属性値は「abc」です
- これらの条件は両方とも true です。
  - foo 属性値は 234-AND-
  - bar 属性値は「bcd」です

これは、次の Kotlin ブール式に相当します。

```
(foo == 123 && bar == "abc") || (foo == 234 && bar == "bcd")
```

関数とプロパティ

次の関数とプロパティは、追加の式機能を提供します。

- contains: 文字列/リスト属性値に指定された値が含まれているかどうかを確認します
- exists: 属性が定義されているかどうかをチェックし、値 (を含むnull) を保持します。
- notExists: 属性が未定義かどうかを確認します
- isOfType: 属性値が文字列、数値、ブール値などの特定のタイプであるかどうかをチェックします。
- size: コレクション内の要素数や文字列の長さなどの属性のサイズを取得します。
- startsWith: 文字列属性値が特定の部分文字列で始まるかどうかを確認します

次の例は、式で使用できる追加の関数とプロパティの使用を示しています。

```
attr("foo") contains "apple" // Specifies that the attribute value `foo` must be
                             // a list that contains an `apple` element or a string
                             // which contains the substring `apple`.
```

```
attr("bar").exists()         // Specifies that the `bar` must exist and have a
                             // value (including potentially `null`).

attr("baz").size lt 100      // Specifies that the attribute value `baz` must have
                             // a size of less than 100.

attr("qux") isOfType AttributeType.String // Specifies that the attribute `qux`
                                          // must have a string value.
```

ソートキーフィルター

ソートキーのフィルター式 (query オペレーションの keyConditionパラメータなど) は、名前付き属性値を使用しません。フィルターでソートキーを使用するには、すべての比較sortKeyでキーワードを使用する必要があります。次の例attr("<sort key name>")に示すように、sortKeyキーワードは を置き換えます。

```
sortKey startsWith "abc" // The sort key attribute value must begin with the
                         // substring `abc`.

sortKey isIn 0..99       // The sort key attribute value must be between 0
                         // and 99 (inclusive).
```

ソートキーフィルターをブールロジックと組み合わせることはできません。また、上記の比較のサブセットのみをサポートします。

- 等価と不等価: すべての比較がサポートされています
- 範囲とセット: サポートされているすべての比較
- ブールロジック: サポートされていません
- 関数とプロパティ: startsWith のみがサポートされています

SDK for Kotlin のコード例

このトピックのコード例は、 で AWS SDK for Kotlin を使用する方法を示しています AWS。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

一部のサービスには、サービス固有のライブラリや関数の活用方法を示す追加のカテゴリ例が含まれています。

サービス

- SDK for Kotlin を使用した API Gateway の例
- SDK for Kotlin を使用した Aurora の例
- SDK for Kotlin を使用した Auto Scaling の例
- SDK for Kotlin を使用した Amazon Bedrock の例
- SDK for Kotlin を使用した Amazon Bedrock ランタイムの例
- SDK for Kotlin を使用した CloudWatch の例
- SDK for Kotlin を使用した CloudWatch Logs のコード例
- SDK for Kotlin を使用する Amazon Cognito ID プロバイダーの例
- SDK for Kotlin を使用した Amazon Comprehend の例
- SDK for Kotlin を使用した DynamoDB の例
- SDK for Kotlin を使用した Amazon EC2 の例
- SDK for Kotlin を使用した Amazon ECR の例
- SDK for Kotlin を使用した OpenSearch Service の例
- SDK for Kotlin を使用した EventBridge の例
- AWS Glue SDK for Kotlin を使用した の例
- SDK for Kotlin を使用した IAM の例
- AWS IoT SDK for Kotlin を使用した の例

- AWS IoT data SDK for Kotlin を使用した の例
- AWS IoT FleetWise SDK for Kotlin を使用した の例
- SDK for Kotlin を使用した Amazon Keyspaces の例
- AWS KMS SDK for Kotlin を使用した の例
- SDK for Kotlin を使用する Lambda の例
- SDK for Kotlin を使用した Amazon Location の例
- SDK for Kotlin を使用した MediaConvert の例
- SDK for Kotlin を使用した Amazon Pinpoint 例
- SDK for Kotlin を使用した Amazon RDS の例
- SDK for Kotlin を使用した Amazon RDS Data Service の例
- SDK for Kotlin を使用した Amazon Redshift の例
- SDK for Kotlin を使用する Amazon Rekognition の例
- SDK for Kotlin を使用した Route 53 ドメイン登録の例
- SDK for Kotlin を使用した Amazon S3 の例
- SDK for Kotlin を使用した SageMaker AI の例
- SDK for Kotlin を使用した Secrets Manager の例
- SDK for Kotlin を使用した Amazon SES の例
- SDK for Kotlin を使用した Amazon SNS の例
- SDK for Kotlin を使用した Amazon SQS の例
- SDK for Kotlin を使用した Step Function の例
- サポート SDK for Kotlin を使用した の例
- SDK for Kotlin を使用した Amazon Translate の例

SDK for Kotlin を使用した API Gateway の例

次のコード例は、API Gateway で AWS SDK for Kotlin 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。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- シナリオ

シナリオ

サーバーレスアプリケーションを作成して写真の管理

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

SDK for Kotlin

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設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、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

SDK for Kotlin を使用した Aurora の例

次のコード例は、Aurora で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、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、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

トピック

- 基本
- アクション
- シナリオ

基本

基本を学ぶ

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.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

This Kotlin example performs the following tasks:

1. Returns a list of the available DB engines.
2. Creates a custom DB parameter group.
3. Gets the parameter groups.
4. Gets the parameters in the group.
5. Modifies the auto_increment_increment parameter.
6. Displays the updated parameter value.
7. Gets a list of allowed engine versions.
8. Creates an Aurora DB cluster 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 the database instance in the cluster to be ready.
13. Creates a snapshot.
14. Waits for DB snapshot to be ready.
15. Deletes the DB instance.
16. Deletes the DB cluster.
17. Deletes the DB cluster group.
 */

var slTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbName> <dbSnapshotIdentifier> <secretName>
        Where:
            dbClusterGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceClusterIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
 the database credentials.
```

```
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val dbClusterGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceClusterIdentifier = args[2]
    val dbInstanceIdentifier = args[3]
    val dbName = args[4]
    val dbSnapshotIdentifier = args[5]
    val secretName = args[6]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeAuroraDBEngines()

    println("2. Create a custom parameter group")
    createDBClusterParameterGroup(dbClusterGroupName, dbParameterGroupFamily)

    println("3. Get the parameter group")
    describeDbClusterParameterGroups(dbClusterGroupName)

    println("4. Get the parameters in the group")
    describeDbClusterParameters(dbClusterGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBClusterParas(dbClusterGroupName)

    println("6. Display the updated parameter value")
    describeDbClusterParameters(dbClusterGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedClusterEngines(dbParameterGroupFamily)

    println("8. Create an Aurora DB cluster database")
```

```
    val arnClusterVal = createDBCluster(dbClusterGroupName, dbName,
 dbInstanceClusterIdentifier, username, userPassword)
    println("The ARN of the cluster is $arnClusterVal")

    println("9. Wait for DB instance to be ready")
    waitForClusterInstanceReady(dbInstanceClusterIdentifier)

    println("10. Get a list of instance classes available for the selected engine")
    val instanceClass = getListInstanceClasses()

    println("11. Create a database instance in the cluster.")
    val clusterDBARN = createDBInstanceCluster(dbInstanceIdentifier,
 dbInstanceClusterIdentifier, instanceClass)
    println("The ARN of the database is $clusterDBARN")

    println("12. Wait for DB instance to be ready")
    waitDBAuroraInstanceReady(dbInstanceIdentifier)

    println("13. Create a snapshot")
    createDBClusterSnapshot(dbInstanceClusterIdentifier, dbSnapshotIdentifier)

    println("14. Wait for DB snapshot to be ready")
    waitSnapshotReady(dbSnapshotIdentifier, dbInstanceClusterIdentifier)

    println("15. Delete the DB instance")
    deleteDBInstance(dbInstanceIdentifier)

    println("16. Delete the DB cluster")
    deleteCluster(dbInstanceClusterIdentifier)

    println("17. Delete the DB cluster group")
    if (clusterDBARN != null) {
        deleteDBClusterGroup(dbClusterGroupName, clusterDBARN)
    }
    println("The Scenario has successfully completed.")
}

@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
```

```
    var instanceARN: String

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}

suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }
```

```
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}

suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}

suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
```

```
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}

suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
```

```
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}

suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

suspend fun getListInstanceClasses(): String {
    val optionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "aurora-mysql"
            maxRecords = 20
        }
    var instanceClass = ""
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeOrderableDbInstanceOptions(optionsRequest)
        response.orderableDbInstanceOptions?.forEach { instanceOption ->
            instanceClass = instanceOption.dbInstanceClass.toString()
```

```
            println("The instance class is ${instanceOption.dbInstanceClass}")
            println("The engine version is ${instanceOption.engineVersion}")
        }
    }
    return instanceClass
}

// Waits until the database instance is available.
suspend fun waitForClusterInstanceReady(dbClusterIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbClustersRequest {
            dbClusterIdentifier = dbClusterIdentifierVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbClusters(instanceRequest)
            response.dbClusters?.forEach { cluster ->
                instanceReadyStr = cluster.status.toString()
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database cluster is available!")
}

suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
```

```
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
```

```
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}

suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
```

```
                }
            }
        }
    }
}

suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}

suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}

suspend fun describeAuroraDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            engine = "aurora-mysql"
```

```
            defaultOnly = true
            maxRecords = 20
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        response.dbEngineVersions?.forEach { engineOb ->
            println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}")
            println("The name of the database engine ${engineOb.engine}")
            println("The version number of the database engine
 ${engineOb.engineVersion}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin 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

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}
```

- API の詳細については、AWS SDK for Kotlin 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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CreateDBClusterParameterGroup」を参照してください。

CreateDBClusterSnapshot

次の例は、CreateDBClusterSnapshot を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CreateDBClusterSnapshot」を参照してください。

`CreateDBInstance`

次の例は、`CreateDBInstance` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}
```

- API の詳細については、AWS SDK for Kotlin 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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
```

```
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

- API の詳細については、AWS SDK for Kotlin 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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
```

```
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DeleteDBClusterParameterGroup」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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin 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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
```

```
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeDBClusterParameterGroups」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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
```

```
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeDBClusterParameters」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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeDBClusterSnapshots」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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
```

```
                println("*** The parameter name is  $paraName")
                println("*** The parameter value is  ${para.parameterValue}")
                println("*** The parameter data type is ${para.dataType}")
                println("*** The parameter description is ${para.description}")
                println("*** The parameter allowed values  is
 ${para.allowedValues}")
            }
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin 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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
```

```
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeDBEngineVersions」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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
```

```
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeDBInstances」を参照してください。

`ModifyDBClusterParameterGroup`

次の例は、`ModifyDBClusterParameterGroup` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
```

```
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ModifyDBClusterParameterGroup」を参照してください。

シナリオ

Aurora Serverless 作業項目トラッカーの作成

次のコード例は、Amazon Aurora Serverless データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信する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。

SDK for Kotlin

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 サーバーレスデータをクエリする Spring REST API と React アプリケーションで使用するための完全なソースコードと設定方法については、GitHub にある完全な例を参照してください。

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

SDK for Kotlin を使用した Auto Scaling の例

次のコード例は、Auto Scaling で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

トピック

- 基本
- アクション

基本

基本を学ぶ

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ーゾーンを使用して、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、リソースをクリーンアップします。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
```

```
        <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>

    Where:
        groupName - The name of the Auto Scaling group.
        launchTemplateName - The name of the launch template.
        serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-linked
role that the Auto Scaling group uses.
        vpcZoneId - A subnet Id for a virtual private cloud (VPC) where instances in
the Auto Scaling group can be created.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val groupName = args[0]
    val launchTemplateName = args[1]
    val serviceLinkedRoleARN = args[2]
    val vpcZoneId = args[3]

    println("**** Create an Auto Scaling group named $groupName")
    createAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN,
vpcZoneId)

    println("Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned")
    delay(60000)

    val instanceId = getSpecificAutoScaling(groupName)
    if (instanceId.compareTo("") == 0) {
        println("Error - no instance Id value")
        exitProcess(1)
    } else {
        println("The instance Id value is $instanceId")
    }

    println("**** Describe Auto Scaling with the Id value $instanceId")
    describeAutoScalingInstance(instanceId)

    println("**** Enable metrics collection $instanceId")
    enableMetricsCollection(groupName)

    println("**** Update an Auto Scaling group to maximum size of 3")
```

```
    updateAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN)

    println("**** Describe all Auto Scaling groups to show the current state of the
 groups")
    describeAutoScalingGroups(groupName)

    println("**** Describe account details")
    describeAccountLimits()

    println("Wait 1 min for the resources, including the instance. Otherwise, an
 empty instance Id is returned")
    delay(60000)

    println("**** Set desired capacity to 2")
    setDesiredCapacity(groupName)

    println("**** Get the two instance Id values and state")
    getAutoScalingGroups(groupName)

    println("**** List the scaling activities that have occurred for the group")
    describeScalingActivities(groupName)

    println("**** Terminate an instance in the Auto Scaling group")
    terminateInstanceInAutoScalingGroup(instanceId)

    println("**** Stop the metrics collection")
    disableMetricsCollection(groupName)

    println("**** Delete the Auto Scaling group")
    deleteSpecificAutoScalingGroup(groupName)
}

suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
```

```
        }
    }
}

suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}

suspend fun describeScalingActivities(groupName: String?) {
    val scalingActivitiesRequest =
        DescribeScalingActivitiesRequest {
            autoScalingGroupName = groupName
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeScalingActivities(scalingActivitiesRequest)
        response.activities?.forEach { activity ->
            println("The activity Id is ${activity.activityId}")
            println("The activity details are ${activity.details}")
        }
    }
}

suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
```

```
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}

suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}

suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
```

```
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
    }
}

suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}
```

```
suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}

suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}

suspend fun getSpecificAutoScaling(groupName: String): String {
    var instanceId = ""
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")

            group.instances?.forEach { instance ->
```

```
                instanceId = instance.instanceId.toString()
            }
        }
    }
    return instanceId
}

suspend fun describeAccountLimits() {
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAccountLimits(DescribeAccountLimitsRequest {})
        println("The max number of Auto Scaling groups is
 ${response.maxNumberOfAutoScalingGroups}")
        println("The current number of Auto Scaling groups is
 ${response.numberOfAutoScalingGroups}")
    }
}

suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}

suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin 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

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
```

```
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateAutoScalingGroup」を参照してください。

`DeleteAutoScalingGroup`

次の例は、`DeleteAutoScalingGroup` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteAutoScalingGroup」を参照してください。

`DescribeAutoScalingGroups`

次の例は、`DescribeAutoScalingGroups` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
```

```
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeAutoScalingGroups」を参照してください。

`DescribeAutoScalingInstances`

次の例は、`DescribeAutoScalingInstances` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeAutoScalingInstances」を参照してください。

`DescribeScalingActivities`

次の例は、`DescribeScalingActivities` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeScalingActivities」を参照してください。

`DisableMetricsCollection`

次の例は、`DisableMetricsCollection` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DisableMetricsCollection」を参照してください。

`EnableMetricsCollection`

次の例は、`EnableMetricsCollection` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
```

```
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「EnableMetricsCollection」を参照してください。

`SetDesiredCapacity`

次の例は、`SetDesiredCapacity` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「SetDesiredCapacity」を参照してください。

`TerminateInstanceInAutoScalingGroup`

次の例は、`TerminateInstanceInAutoScalingGroup` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「TerminateInstanceInAutoScalingGroup」を参照してください。

`UpdateAutoScalingGroup`

次の例は、`UpdateAutoScalingGroup` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「UpdateAutoScalingGroup」を参照してください。

SDK for Kotlin を使用した Amazon Bedrock の例

次のコード例は、Amazon Bedrock で AWS SDK for Kotlin 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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- アクション

アクション

`ListFoundationModels`

次の例は、`ListFoundationModels` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

利用可能な Amazon Bedrock 基盤モデルを一覧表示します。

```
suspend fun listFoundationModels(): List<FoundationModelSummary>? {
    BedrockClient { region = "us-east-1" }.use { bedrockClient ->
        val response =
 bedrockClient.listFoundationModels(ListFoundationModelsRequest {})
        response.modelSummaries?.forEach { model ->
            println("========================================")
            println(" Model ID: ${model.modelId}")
            println("----------------------------------------")
```

```
            println(" Name: ${model.modelName}")
            println(" Provider: ${model.providerName}")
            println(" Input modalities: ${model.inputModalities}")
            println(" Output modalities: ${model.outputModalities}")
            println(" Supported customizations: ${model.customizationsSupported}")
            println(" Supported inference types: ${model.inferenceTypesSupported}")
            println("----------------------------------------\n")
        }
        return response.modelSummaries
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ListFoundationModels」を参照してください。

SDK for Kotlin を使用した Amazon Bedrock ランタイムの例

次のコード例は、Amazon Bedrock ランタイムで AWS SDK for Kotlin 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。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- Amazon Nova
- Amazon Titan Text

Amazon Nova

Converse

次のコード例は、Bedrock の Converse API を使用して Amazon Nova 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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 Nova にテキストメッセージを送信します。

```
import aws.sdk.kotlin.services.bedrockruntime.BedrockRuntimeClient
import aws.sdk.kotlin.services.bedrockruntime.model.ContentBlock
import aws.sdk.kotlin.services.bedrockruntime.model.ConversationRole
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseRequest
import aws.sdk.kotlin.services.bedrockruntime.model.Message

/**
 * This example demonstrates how to use the Amazon Nova foundation models to
 generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
suspend fun main() {
    converse().also { println(it) }
}

suspend fun converse(): String {
    // Create and configure the Bedrock runtime client
    BedrockRuntimeClient { region = "us-east-1" }.use { client ->

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        val modelId = "amazon.nova-lite-v1:0"

        // Create the message with the user's prompt
        val prompt = "Describe the purpose of a 'hello world' program in one line."
        val message = Message {
            role = ConversationRole.User
```

```
            content = listOf(ContentBlock.Text(prompt))
        }

        // Configure the request with optional model parameters
        val request = ConverseRequest {
            this.modelId = modelId
            messages = listOf(message)
            inferenceConfig {
                maxTokens = 500 // Maximum response length
                temperature = 0.5F // Lower values: more focused output
                // topP = 0.8F // Alternative to temperature
            }
        }

        // Send the request and process the model's response
        runCatching {
            val response = client.converse(request)
            return response.output!!.asMessage().content.first().asText()
        }.getOrElse { error ->
            error.message?.let { e -> System.err.println("ERROR: Can't invoke
 '$modelId'. Reason: $e") }
            throw RuntimeException("Failed to generate text with model $modelId",
 error)
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin の API リファレンス」の「Converse」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 Nova 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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 Nova 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。

```
import aws.sdk.kotlin.services.bedrockruntime.BedrockRuntimeClient
import aws.sdk.kotlin.services.bedrockruntime.model.ContentBlock
import aws.sdk.kotlin.services.bedrockruntime.model.ConversationRole
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseStreamOutput
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseStreamRequest
import aws.sdk.kotlin.services.bedrockruntime.model.Message

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message with a prompt
 * - Configure a streaming request with parameters
 * - Process the response stream in real time
 */
suspend fun main() {
    converseStream()
}

suspend fun converseStream(): String {
    // A buffer to collect the complete response
    val completeResponseBuffer = StringBuilder()

    // Create and configure the Bedrock runtime client
    BedrockRuntimeClient { region = "us-east-1" }.use { client ->

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
```

```
        val modelId = "amazon.nova-lite-v1:0"

        // Create the message with the user's prompt
        val prompt = "Describe the purpose of a 'hello world' program in a
paragraph."
        val message = Message {
            role = ConversationRole.User
            content = listOf(ContentBlock.Text(prompt))
        }

        // Configure the request with optional model parameters
        val request = ConverseStreamRequest {
            this.modelId = modelId
            messages = listOf(message)
            inferenceConfig {
                maxTokens = 500 // Maximum response length
                temperature = 0.5F // Lower values: more focused output
                // topP = 0.8F // Alternative to temperature
            }
        }

        // Process the streaming response
        runCatching {
            client.converseStream(request) { response ->
                response.stream?.collect { chunk ->
                    when (chunk) {
                        is ConverseStreamOutput.ContentBlockDelta -> {
                            // Process each text chunk as it arrives
                            chunk.value.delta?.asText()?.let { text ->
                                print(text)
                                System.out.flush() // Ensure immediate output
                                completeResponseBuffer.append(text)
                            }
                        }
                        else -> {} // Other output block types can be handled as
needed
                    }
                }
            }
        }.onFailure { error ->
            error.message?.let { e -> System.err.println("ERROR: Can't invoke
'$modelId'. Reason: $e") }
            throw RuntimeException("Failed to generate text with model $modelId:
$error", error)
```

```
        }
    }

    return completeResponseBuffer.toString()
}
```

- API の詳細については、「AWS SDK for Kotlin の API リファレンス」の「ConverseStream」を参照してください。

Amazon Titan Text

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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 を使用してショートストーリーを生成します。

```
import aws.sdk.kotlin.services.bedrockruntime.BedrockRuntimeClient
import aws.sdk.kotlin.services.bedrockruntime.model.InvokeModelRequest
import kotlinx.serialization.Serializable
import kotlinx.serialization.json.Json

/**
 * This example demonstrates how to use the Amazon Titan foundation models to
 generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a request payload
 * - Configure and send a request
```

```
 * - Process the response
 */
suspend fun main() {
    invokeModel().also { println(it) }
}

// Data class for parsing the model's response
@Serializable
private data class BedrockResponse(val results: List<Result>) {
    @Serializable
    data class Result(
        val outputText: String,
    )
}

// Initialize JSON parser with relaxed configuration
private val json = Json { ignoreUnknownKeys = true }

suspend fun invokeModel(): String {
    // Create and configure the Bedrock runtime client
    BedrockRuntimeClient { region = "us-east-1" }.use { client ->

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        val modelId = "amazon.titan-text-lite-v1"

        // Create the request payload with optional configuration parameters
        // For detailed parameter descriptions, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-text.html
        val prompt = "Describe the purpose of a 'hello world' program in one line."
        val request = """
            {
                "inputText": "$prompt",
                "textGenerationConfig": {
                    "maxTokenCount": 500,
                    "temperature": 0.5
                }
            }
        """.trimIndent()

        // Send the request and process the model's response
        runCatching {
```

```
            // Send the request to the model
            val response = client.invokeModel(
                InvokeModelRequest {
                    this.modelId = modelId
                    body = request.toByteArray()
                },
            )

            // Convert the response bytes to a JSON string
            val jsonResponse = response.body.toString(Charsets.UTF_8)

            // Parse the JSON into a Kotlin object
            val parsedResponse =
 json.decodeFromString<BedrockResponse>(jsonResponse)

            // Extract and return the generated text
            return parsedResponse.results.firstOrNull()!!.outputText
        }.getOrElse { error ->
            error.message?.let { msg ->
                System.err.println("ERROR: Can't invoke '$modelId'. Reason: $msg")
            }
            throw RuntimeException("Failed to generate text with model $modelId",
 error)
        }
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API InvokeModel」を参照してください。

SDK for Kotlin を使用した CloudWatch の例

次のコード例は、CloudWatch で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

Hello CloudWatch

次のコード例は、CloudWatch の使用を開始する方法を示しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
           <namespace>
        Where:
           namespace - The namespace to filter against (for example, AWS/EC2).
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val namespace = args[0]
    listAllMets(namespace)
}
```

```
suspend fun listAllMets(namespaceVal: String?) {
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listMetricsPaginated(request)
            .transform { it.metrics?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.metricName}")
                println("Namespace is ${obj.namespace}")
            }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ListMetrics」を参照してください。

トピック

- 基本
- アクション

基本

基本を学ぶ

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 の名前空間とメトリクスをリストする。
- 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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

CloudWatch の機能を実証するインタラクティブなシナリオを実行します。

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.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 Kotlin code example performs the following tasks:

 1. List available namespaces from Amazon CloudWatch. Select a namespace from the
 list.
 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.
 */

val DASHES: String? = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <myDate> <costDateWeek> <dashboardName> <dashboardJson> <dashboardAdd>
 <settings> <metricImage>

        Where:
            myDate - The start date to use to get metric statistics. (For example,
 2023-01-11T18:35:24.00Z.)
            costDateWeek - The start date to use to get AWS Billing and Cost
 Management statistics. (For example, 2023-01-11T18:35:24.00Z.)
            dashboardName - The name of the dashboard to create.
            dashboardJson - The location of a JSON file to use to create a
 dashboard. (See Readme file.)
            dashboardAdd - The location of a JSON file to use to update a dashboard.
 (See Readme file.)
            settings - The location of a JSON file from which various values are
 read. (See Readme file.)
            metricImage - The location of a BMP file that is used to create a
 graph.
    """

    if (args.size != 7) {
        println(usage)
        System.exit(1)
    }

    val myDate = args[0]
    val costDateWeek = args[1]
    val dashboardName = args[2]
    val dashboardJson = args[3]
    val dashboardAdd = args[4]
    val settings = args[5]
    var metricImage = args[6]
    val dataPoint = "10.0".toDouble()
    val inOb = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the Amazon CloudWatch example scenario.")
```

```
    println(DASHES)

    println(DASHES)
    println("1. List at least five available unique namespaces from Amazon
CloudWatch. Select a CloudWatch namespace from the list.")
    val list: ArrayList<String> = listNameSpaces()
    for (z in 0..4) {
        println("    ${z + 1}. ${list[z]}")
    }

    var selectedNamespace: String
    var selectedMetrics = ""
    var num = inOb.nextLine().toInt()
    println("You selected $num")

    if (1 <= num && num <= 5) {
        selectedNamespace = list[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $selectedNamespace")
    println(DASHES)

    println(DASHES)
    println("2. List available metrics within the selected namespace and select one
from the list.")
    val metList = listMets(selectedNamespace)
    for (z in 0..4) {
        println("    ${ z + 1}. ${metList?.get(z)}")
    }
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        selectedMetrics = metList![num - 1]
    } else {
        println("You did not select a valid option.")
        System.exit(1)
    }
    println("You selected $selectedMetrics")
    val myDimension = getSpecificMet(selectedNamespace)
    if (myDimension == null) {
        println("Error - Dimension is null")
        exitProcess(1)
    }
```

```
    println(DASHES)

    println(DASHES)
    println("3. Get statistics for the selected metric over the last day.")
    val metricOption: String
    val statTypes = ArrayList<String>()
    statTypes.add("SampleCount")
    statTypes.add("Average")
    statTypes.add("Sum")
    statTypes.add("Minimum")
    statTypes.add("Maximum")

    for (t in 0..4) {
        println("    ${t + 1}. ${statTypes[t]}")
    }
    println("Select a metric statistic by entering a number from the preceding
list:")
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        metricOption = statTypes[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $metricOption")
    getAndDisplayMetricStatistics(selectedNamespace, selectedMetrics, metricOption,
myDate, myDimension)
    println(DASHES)

    println(DASHES)
    println("4. Get CloudWatch estimated billing for the last week.")
    getMetricStatistics(costDateWeek)
    println(DASHES)

    println(DASHES)
    println("5. Create a new CloudWatch dashboard with metrics.")
    createDashboardWithMetrics(dashboardName, dashboardJson)
    println(DASHES)

    println(DASHES)
    println("6. List dashboards using a paginator.")
    listDashboards()
    println(DASHES)
```

```
    println(DASHES)
    println("7. Create a new custom metric by adding data to it.")
    createNewCustomMetric(dataPoint)
    println(DASHES)

    println(DASHES)
    println("8. Add an additional metric to the dashboard.")
    addMetricToDashboard(dashboardAdd, dashboardName)
    println(DASHES)

    println(DASHES)
    println("9. Create an alarm for the custom metric.")
    val alarmName: String = createAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("10. Describe 10 current alarms.")
    describeAlarms()
    println(DASHES)

    println(DASHES)
    println("11. Get current data for the new custom metric.")
    getCustomMetricData(settings)
    println(DASHES)

    println(DASHES)
    println("12. Push data into the custom metric to trigger the alarm.")
    addMetricDataForAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("13. Check the alarm state using the action DescribeAlarmsForMetric.")
    checkForMetricAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("14. Get alarm history for the new alarm.")
    getAlarmHistory(settings, myDate)
    println(DASHES)

    println(DASHES)
    println("15. Add an anomaly detector for the custom metric.")
    addAnomalyDetector(settings)
    println(DASHES)
```

```
    println(DASHES)
    println("16. Describe current anomaly detectors.")
    describeAnomalyDetectors(settings)
    println(DASHES)

    println(DASHES)
    println("17. Get a metric image for the custom metric.")
    getAndOpenMetricImage(metricImage)
    println(DASHES)

    println(DASHES)
    println("18. Clean up the Amazon CloudWatch resources.")
    deleteDashboard(dashboardName)
    deleteAlarm(alarmName)
    deleteAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("The Amazon CloudWatch example scenario is complete.")
    println(DASHES)
}

suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val request =
        DeleteAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
```

```
        println("Successfully deleted the Anomaly Detector.")
    }
}

suspend fun deleteAlarm(alarmNameVal: String) {
    val request =
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}

suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}

suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
```

```
        }"""

    val imageRequest =
        GetMetricWidgetImageRequest {
            metricWidget = myJSON
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}

suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest =
        DescribeAnomalyDetectorsRequest {
            maxResults = 10
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}

suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
```

```
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val anomalyDetectorRequest =
        PutAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}

suspend fun getAlarmHistory(
    fileName: String,
    date: String,
) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest =
        DescribeAlarmHistoryRequest {
            startDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDateVal)
            alarmName = alarmNameVal
            historyItemType = HistoryItemType.Action
        }
```

```
    CloudWatchClient {
        credentialsProvider = EnvironmentCredentialsProvider()
        region = "us-east-1"
    }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}

suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest =
        DescribeAlarmsForMetricRequest {
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) {
```

```
            println("No Alarm state found for $customMetricName after 10 retries.")
        } else {
            println("Alarm state found for $customMetricName.")
        }
    }
}

suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1001.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val datum2 =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1002.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)

    val request =
        PutMetricDataRequest {
```

```
            namespace = customMetricNamespace
            metricData = metricDataList
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}

suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 =
        nowDate.plus(hours, ChronoUnit.HOURS).plus(
            minutes,
            ChronoUnit.MINUTES,
        )

    val met =
        Metric {
            metricName = customMetricName
            namespace = customMetricNamespace
        }

    val metStat =
        MetricStat {
            stat = "Maximum"
            period = 1
            metric = met
        }

    val dataQUery =
        MetricDataQuery {
            metricStat = metStat
            id = "foo2"
```

```
            returnData = true
        }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq =
        GetMetricDataRequest {
            maxDatapoints = 10
            scanBy = ScanBy.TimestampDescending
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(nowDate)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(date2)
            metricDataQueries = dq
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}

suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
            alarmTypes = typeList
            maxRecords = 10
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}
```

```
suspend fun createAlarm(fileName: String): String {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode: JsonNode = ObjectMapper().readTree(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val emailTopic = rootNode.findValue("emailTopic").asText()
    val accountId = rootNode.findValue("accountId").asText()
    val region2 = rootNode.findValue("region").asText()

    // Create a List for alarm actions.
    val alarmActionObs: MutableList<String> = ArrayList()
    alarmActionObs.add("arn:aws:sns:$region2:$accountId:$emailTopic")
    val alarmRequest =
        PutMetricAlarmRequest {
            alarmActions = alarmActionObs
            alarmDescription = "Example metric alarm"
            alarmName = alarmNameVal
            comparisonOperator = ComparisonOperator.GreaterThanOrEqualToThreshold
            threshold = 100.00
            metricName = customMetricName
            namespace = customMetricNamespace
            evaluationPeriods = 1
            period = 10
            statistic = Statistic.Maximum
            datapointsToAlarm = 1
            treatMissingData = "ignore"
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(alarmRequest)
        println("$alarmNameVal was successfully created!")
        return alarmNameVal
    }
}

suspend fun addMetricToDashboard(
    fileNameVal: String,
    dashboardNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
```

```
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully updated.")
    }
}

suspend fun createNewCustomMetric(dataPoint: Double) {
    val dimension =
        Dimension {
            name = "UNIQUE_PAGES"
            value = "URLS"
        }

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = "PAGES_VISITED"
            unit = StandardUnit.None
            value = dataPoint
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
            dimensions = listOf(dimension)
        }

    val request =
        PutMetricDataRequest {
            namespace = "SITE/TRAFFIC"
            metricData = listOf(datum)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric PAGES_VISITED")
    }
}
```

```
suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}

suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}

fun readFileAsString(file: String): String =
 String(Files.readAllBytes(Paths.get(file)))

suspend fun getMetricStatistics(costDateWeek: String?) {
    val start = Instant.parse(costDateWeek)
    val endDate = Instant.now()
```

```
    val dimension =
        Dimension {
            name = "Currency"
            value = "USD"
        }

    val dimensionList: MutableList<Dimension> = ArrayList()
    dimensionList.add(dimension)

    val statisticsRequest =
        GetMetricStatisticsRequest {
            metricName = "EstimatedCharges"
            namespace = "AWS/Billing"
            dimensions = dimensionList
            statistics = listOf(Statistic.Maximum)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            period = 86400
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data: List<Datapoint>? = response.datapoints
        if (data != null) {
            if (!data.isEmpty()) {
                for (datapoint in data) {
                    println("Timestamp:  ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

suspend fun getAndDisplayMetricStatistics(
    nameSpaceVal: String,
    metVal: String,
    metricOption: String,
    date: String,
```

```
    myDimension: Dimension,
) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest =
        GetMetricStatisticsRequest {
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            dimensions = listOf(myDimension)
            metricName = metVal
            namespace = nameSpaceVal
            period = 86400
            statistics = listOf(Statistic.fromValue(metricOption))
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
```

```
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}

suspend fun getSpecificMet(namespaceVal: String?): Dimension? {
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(request)
        val myList = response.metrics
        if (myList != null) {
            return myList[0].dimensions?.get(0)
        }
    }
    return null
}

suspend fun listNameSpaces(): ArrayList<String> {
    val nameSpaceList = ArrayList<String>()
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(ListMetricsRequest {})
        response.metrics?.forEach { metrics ->
            val data = metrics.namespace
            if (!nameSpaceList.contains(data)) {
                nameSpaceList.add(data!!)
            }
        }
    }
    return nameSpaceList
}
```

- API の詳細については、「AWS SDK for Kotlin 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

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、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。

```
suspend fun deleteAlarm(alarmNameVal: String) {
    val request =
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
```

```
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DeleteAlarms」を参照してください。

`DeleteAnomalyDetector`

次の例は、`DeleteAnomalyDetector` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val request =
        DeleteAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
```

```
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteAnomalyDetector」を参照してください。

`DeleteDashboards`

次の例は、`DeleteDashboards` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteDashboards」を参照してください。

`DescribeAlarmHistory`

次の例は、`DescribeAlarmHistory` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getAlarmHistory(
    fileName: String,
    date: String,
) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest =
        DescribeAlarmHistoryRequest {
            startDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDateVal)
            alarmName = alarmNameVal
            historyItemType = HistoryItemType.Action
        }

    CloudWatchClient {
        credentialsProvider = EnvironmentCredentialsProvider()
        region = "us-east-1"
    }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
```

```
                println("History summary ${item.historySummary}")
                println("Time stamp: ${item.timestamp}")
            }
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeAlarmHistory」を参照してください。

`DescribeAlarms`

次の例は、`DescribeAlarms` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
            alarmTypes = typeList
            maxRecords = 10
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeAlarms」を参照してください。

`DescribeAlarmsForMetric`

次の例は、`DescribeAlarmsForMetric` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、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。

```
suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest =
        DescribeAlarmsForMetricRequest {
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
```

```
        if (!hasAlarm) {
            println("No Alarm state found for $customMetricName after 10 retries.")
        } else {
            println("Alarm state found for $customMetricName.")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeAlarmsForMetric」を参照してください。

`DescribeAnomalyDetectors`

次の例は、`DescribeAnomalyDetectors` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest =
        DescribeAnomalyDetectorsRequest {
            maxResults = 10
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
```

```
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeAnomalyDetectors」を参照してください。

`DisableAlarmActions`

次の例は、`DisableAlarmActions` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、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。

```
suspend fun disableActions(alarmName: String) {
    val request =
        DisableAlarmActionsRequest {
            alarmNames = listOf(alarmName)
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.disableAlarmActions(request)
        println("Successfully disabled actions on alarm $alarmName")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DisableAlarmActions」を参照してください。

`EnableAlarmActions`

次の例は、`EnableAlarmActions` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、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。

```
suspend fun enableActions(alarm: String) {
    val request =
        EnableAlarmActionsRequest {
            alarmNames = listOf(alarm)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.enableAlarmActions(request)
        println("Successfully enabled actions on alarm $alarm")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「EnableAlarmActions」を参照してください。

`GetMetricData`

次の例は、`GetMetricData` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 =
        nowDate.plus(hours, ChronoUnit.HOURS).plus(
            minutes,
            ChronoUnit.MINUTES,
        )

    val met =
        Metric {
            metricName = customMetricName
            namespace = customMetricNamespace
        }

    val metStat =
        MetricStat {
            stat = "Maximum"
            period = 1
            metric = met
        }

    val dataQUery =
        MetricDataQuery {
            metricStat = metStat
            id = "foo2"
            returnData = true
        }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq =
        GetMetricDataRequest {
            maxDatapoints = 10
            scanBy = ScanBy.TimestampDescending
```

```
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(nowDate)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(date2)
            metricDataQueries = dq
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetMetricData」を参照してください。

`GetMetricStatistics`

次の例は、`GetMetricStatistics` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getAndDisplayMetricStatistics(
    nameSpaceVal: String,
    metVal: String,
    metricOption: String,
    date: String,
    myDimension: Dimension,
) {
```

```
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest =
        GetMetricStatisticsRequest {
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            dimensions = listOf(myDimension)
            metricName = metVal
            namespace = nameSpaceVal
            period = 86400
            statistics = listOf(Statistic.fromValue(metricOption))
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetMetricStatistics」を参照してください。

GetMetricWidgetImage

次の例は、GetMetricWidgetImage を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
        }"""

    val imageRequest =
        GetMetricWidgetImageRequest {
            metricWidget = myJSON
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetMetricWidgetImage」を参照してください。

`ListDashboards`

次の例は、`ListDashboards` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ListDashboards」を参照してください。

`ListMetrics`

次の例は、`ListMetrics` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ListMetrics」を参照してください。

`PutAnomalyDetector`

次の例は、`PutAnomalyDetector` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val anomalyDetectorRequest =
        PutAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「PutAnomalyDetector」を参照してください。

`PutDashboard`

次の例は、`PutDashboard` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「PutDashboard」を参照してください。

`PutMetricAlarm`

次の例は、`PutMetricAlarm` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、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。

```
suspend fun putMetricAlarm(
    alarmNameVal: String,
    instanceIdVal: String,
) {
    val dimensionOb =
        Dimension {
            name = "InstanceId"
            value = instanceIdVal
        }

    val request =
        PutMetricAlarmRequest {
            alarmName = alarmNameVal
            comparisonOperator = ComparisonOperator.GreaterThanThreshold
            evaluationPeriods = 1
            metricName = "CPUUtilization"
            namespace = "AWS/EC2"
            period = 60
            statistic = Statistic.fromValue("Average")
            threshold = 70.0
            actionsEnabled = false
            alarmDescription = "An Alarm created by the Kotlin SDK when server CPU
 utilization exceeds 70%"
            unit = StandardUnit.fromValue("Seconds")
            dimensions = listOf(dimensionOb)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(request)
        println("Successfully created an alarm with name $alarmNameVal")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「PutMetricAlarm」を参照してください。

PutMetricData

次の例は、PutMetricData を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1001.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val datum2 =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1002.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)
```

```
    val request =
        PutMetricDataRequest {
            namespace = customMetricNamespace
            metricData = metricDataList
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「PutMetricData」を参照してください。

SDK for Kotlin を使用した CloudWatch Logs のコード例

次のコード例は、CloudWatch Logs で AWS SDK for Kotlin 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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteSubFilter(
    filter: String?,
    logGroup: String?,
) {
    val request =
        DeleteSubscriptionFilterRequest {
            filterName = filter
            logGroupName = logGroup
        }

    CloudWatchLogsClient { region = "us-west-2" }.use { logs ->
        logs.deleteSubscriptionFilter(request)
        println("Successfully deleted CloudWatch logs subscription filter named
 $filter")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteSubscriptionFilter」を参照してください。

`DescribeSubscriptionFilters`

次の例は、`DescribeSubscriptionFilters` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeFilters(logGroup: String) {
    val request =
        DescribeSubscriptionFiltersRequest {
            logGroupName = logGroup
            limit = 1
        }

    CloudWatchLogsClient { region = "us-west-2" }.use { cwlClient ->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name  ${filter.filterName} pattern
 ${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeSubscriptionFilters」を参照してください。

`StartLiveTail`

次の例は、StartLiveTail を使用する方法を説明しています。

SDK for Kotlin

必要なファイルを含めます。

```
import aws.sdk.kotlin.services.cloudwatchlogs.CloudWatchLogsClient
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailRequest
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailResponseStream
import kotlinx.coroutines.flow.takeWhile
```

Live Tail セッションを開始します。

```
    val client = CloudWatchLogsClient.fromEnvironment()

    val request = StartLiveTailRequest {
        logGroupIdentifiers = logGroupIdentifiersVal
        logStreamNames = logStreamNamesVal
```

```
        logEventFilterPattern = logEventFilterPatternVal
    }

    val startTime = System.currentTimeMillis()

    try {
        client.startLiveTail(request) { response ->
            val stream = response.responseStream
            if (stream != null) {
                /* Set a timeout to unsubcribe from the flow. This will:
                * 1). Close the stream
                * 2). Stop the Live Tail session
                */
                stream.takeWhile { System.currentTimeMillis() - startTime <
10000 }.collect { value ->
                    if (value is StartLiveTailResponseStream.SessionStart) {
                        println(value.asSessionStart())
                    } else if (value is StartLiveTailResponseStream.SessionUpdate) {
                        for (e in value.asSessionUpdate().sessionResults!!) {
                            println(e)
                        }
                    } else {
                        throw IllegalArgumentException("Unknown event type")
                    }
                }
            } else {
                throw IllegalArgumentException("No response stream")
            }
        }
    } catch (e: Exception) {
        println("Exception occurred during StartLiveTail: $e")
        System.exit(1)
    }
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「StartLiveTail」を参照してください。

SDK for Kotlin を使用する Amazon Cognito ID プロバイダーの例

次のコード例は、Amazon Cognito ID プロバイダーで AWS SDK for Kotlin 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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する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

`AdminGetUser`

次の例は、`AdminGetUser` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
```

```
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「AdminGetUser」を参照してください。

`AdminInitiateAuth`

次の例は、`AdminInitiateAuth` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
```

```
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「AdminInitiateAuth」を参照してください。

`AdminRespondToAuthChallenge`

次の例は、`AdminRespondToAuthChallenge` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }
```

```
    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「AdminRespondToAuthChallenge」を参照してください。

`AssociateSoftwareToken`

次の例は、`AssociateSoftwareToken` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
```

```
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「AssociateSoftwareToken」を参照してください。

`ConfirmSignUp`

次の例は、`ConfirmSignUp` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ConfirmSignUp」を参照してください。

`ListUsers`

次の例は、`ListUsers` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listAllUsers(userPoolId: String) {
    val request =
        ListUsersRequest {
            this.userPoolId = userPoolId
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ListUsers」を参照してください。

`ResendConfirmationCode`

次の例は、`ResendConfirmationCode` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ResendConfirmationCode」を参照してください。

`SignUp`

次の例は、SignUp を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
```

```
        AttributeType {
            name = "email"
            value = emailVal
        }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「SignUp」を参照してください。

`VerifySoftwareToken`

次の例は、`VerifySoftwareToken` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
```

```
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「VerifySoftwareToken」を参照してください。

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
```

```
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.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 initiateAuth 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.
 */

suspend fun main(args: Array<String>) {
    val 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.
    """

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    val clientId = args[0]
    val poolId = args[1]
```

```
    // Use the console to get data from the user.
    println("*** Enter your use name")
    val inOb = Scanner(System.`in`)
    val userName = inOb.nextLine()
    println(userName)

    println("*** Enter your password")
    val password: String = inOb.nextLine()

    println("*** Enter your email")
    val email = inOb.nextLine()

    println("*** Signing up $userName")
    signUp(clientId, userName, password, email)

    println("*** Getting $userName in the user pool")
    getAdminUser(userName, poolId)

    println("*** Conformation code sent to $userName. Would you like to send a new
code? (Yes/No)")
    val ans = inOb.nextLine()

    if (ans.compareTo("Yes") == 0) {
        println("*** Sending a new confirmation code")
        resendConfirmationCode(clientId, userName)
    }
    println("*** Enter the confirmation code that was emailed")
    val code = inOb.nextLine()
    confirmSignUp(clientId, code, userName)

    println("*** Rechecking the status of $userName in the user pool")
    getAdminUser(userName, poolId)

    val authResponse = checkAuthMethod(clientId, userName, password, poolId)
    val mySession = authResponse.session
    val newSession = getSecretForAppMFA(mySession)
    println("*** Enter the 6-digit code displayed in Google Authenticator")
    val myCode = inOb.nextLine()

    // Verify the TOTP and register for MFA.
    verifyTOTP(newSession, myCode)
    println("*** Re-enter a 6-digit code displayed in Google Authenticator")
    val mfaCode: String = inOb.nextLine()
```

```
    val authResponse1 = checkAuthMethod(clientId, userName, password, poolId)
    val session2 = authResponse1.session
    adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}

suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}

suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
```

```
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}

// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }
```

```
    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}

suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}

suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

```
suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}

suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
            name = "email"
            value = emailVal
        }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
```

```
    }
}
```

- API の詳細については、「AWS SDK for Kotlin 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 Kotlin を使用した Amazon Comprehend の例

次のコード例は、Amazon Comprehend で AWS SDK for Kotlin 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。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- シナリオ

シナリオ

Slack メッセージングアプリケーションを作成します。

次のコード例は、Amazon SQS を使用してメッセージングアプリケーションを作成する方法を示しています。

SDK for Kotlin

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および実行の手順については、GitHub で完全な例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
- Amazon SQS

SDK for Kotlin を使用した DynamoDB の例

次のコード例は、DynamoDB で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、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、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

トピック

- 基本
- アクション

- シナリオ

基本

基本を学ぶ

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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。

```
suspend fun createScenarioTable(
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
```

```
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->

        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
 ${response.tableDescription?.tableArn}")
    }
}
```

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.
suspend fun loadData(
    tableName: String,
    fileName: String,
) {
```

```
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}
```

テーブルから項目を取得します。

```
suspend fun getMovie(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

完全な例です。

```
suspend fun main() {
    val tableName = "Movies"
    val fileName = "../../../resources/sample_files/movies.json"
    val partitionAlias = "#a"

    println("Creating an Amazon DynamoDB table named Movies with a key named id and
 a sort key named title.")
    createScenarioTable(tableName, "year")
    loadData(tableName, fileName)
    getMovie(tableName, "year", "1933")
    scanMovies(tableName)
```

```
    val count = queryMovieTable(tableName, "year", partitionAlias)
    println("There are $count Movies released in 2013.")
    deletIssuesTable(tableName)
}

suspend fun createScenarioTable(
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->

        val response = ddb.createTable(request)
```

```
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
 ${response.tableDescription?.tableArn}")
    }
}

// Load data into the table.
suspend fun loadData(
    tableName: String,
    fileName: String,
) {
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
```

```
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}

suspend fun getMovie(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}

suspend fun deletIssuesTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
```

```
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun queryMovieTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionAlias: String,
): Int {
    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = "year"

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.N("2013")

    val request =
        QueryRequest {
            tableName = tableNameVal
            keyConditionExpression = "$partitionAlias = :$partitionKeyName"
            expressionAttributeNames = attrNameAlias
            this.expressionAttributeValues = attrValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}

suspend fun scanMovies(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
```

```
                println("The key name is $key\n")
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin 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

アクション

`CreateTable`

次の例は、`CreateTable` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createNewTable(
    tableNameVal: String,
```

```
    key: String,
): String? {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef)
            keySchema = listOf(keySchemaVal)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        var tableArn: String
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        tableArn = response.tableDescription!!.tableArn.toString()
        println("Table $tableArn is ready")
        return tableArn
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateTable」を参照してください。

`DeleteItem`

次の例は、`DeleteItem` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteDynamoDBItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request =
        DeleteItemRequest {
            tableName = tableNameVal
            key = keyToGet
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteItem(request)
        println("Item with key matching $keyVal was deleted")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteItem」を参照してください。

`DeleteTable`

次の例は、`DeleteTable` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteDynamoDBTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- API の詳細については、『AWS SDK for Kotlin API リファレンス』の「DeleteTable」を参照してください。

`GetItem`

次の例は、`GetItem` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getSpecificItem(
    tableNameVal: String,
    keyName: String,
```

```
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetItem」を参照してください。

`ListTables`

次の例は、`ListTables` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listAllTables() {
    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.listTables(ListTablesRequest {})
        response.tableNames?.forEach { tableName ->
```

```
            println("Table name is $tableName")
        }
    }
}
```

- API の詳細については、『AWS SDK for Kotlin API リファレンス』の「ListTables」を参照してください。

`PutItem`

次の例は、`PutItem` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun putItemInTable(
    tableNameVal: String,
    key: String,
    keyVal: String,
    albumTitle: String,
    albumTitleValue: String,
    awards: String,
    awardVal: String,
    songTitle: String,
    songTitleVal: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()

    // Add all content to the table.
    itemValues[key] = AttributeValue.S(keyVal)
    itemValues[songTitle] = AttributeValue.S(songTitleVal)
    itemValues[albumTitle] = AttributeValue.S(albumTitleValue)
    itemValues[awards] = AttributeValue.S(awardVal)

    val request =
```

```
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println(" A new item was placed into $tableNameVal.")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「PutItem」を参照してください。

`Query`

次の例は、`Query` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun queryDynTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionKeyVal: String,
    partitionAlias: String,
): Int {
    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = partitionKeyName

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.S(partitionKeyVal)

    val request =
```

```
        QueryRequest {
            tableName = tableNameVal
            keyConditionExpression = "$partitionAlias = :$partitionKeyName"
            expressionAttributeNames = attrNameAlias
            this.expressionAttributeValues = attrValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}
```

- API の詳細については、「AWS SDK for Kotlin 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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun scanItems(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
                println("The value is ${item[key]}")
```

```
            }
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin 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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun updateTableItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
    name: String,
    updateVal: String,
) {
    val itemKey = mutableMapOf<String, AttributeValue>()
    itemKey[keyName] = AttributeValue.S(keyVal)

    val updatedValues = mutableMapOf<String, AttributeValueUpdate>()
    updatedValues[name] =
        AttributeValueUpdate {
            value = AttributeValue.S(updateVal)
            action = AttributeAction.Put
        }

    val request =
        UpdateItemRequest {
            tableName = tableNameVal
```

```
            key = itemKey
            attributeUpdates = updatedValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.updateItem(request)
        println("Item in $tableNameVal was updated")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「UpdateItem」を参照してください。

シナリオ

サーバーレスアプリケーションを作成して写真の管理

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

SDK for Kotlin

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設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、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

次のコード例は、Amazon DynamoDB テーブル内の作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信する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。

SDK for Kotlin

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および実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- DynamoDB
- Amazon SES

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun main() {
    val ddb = DynamoDbClient { region = "us-east-1" }
    val tableName = "MoviesPartiQBatch"
```

```
    println("Creating an Amazon DynamoDB table named $tableName with a key named id
 and a sort key named title.")
    createTablePartiQLBatch(ddb, tableName, "year")
    putRecordBatch(ddb)
    updateTableItemBatchBatch(ddb)
    deleteItemsBatch(ddb)
    deleteTablePartiQLBatch(tableName)
}

suspend fun createTablePartiQLBatch(
    ddb: DynamoDbClient,
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
```

```
        }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists {
        // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}

suspend fun putRecordBatch(ddb: DynamoDbClient) {
    val sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?, 'title' : ?,
 'info' : ?}"

    // Create three movies to add to the Amazon DynamoDB table.
    val parametersMovie1 = mutableListOf<AttributeValue>()
    parametersMovie1.add(AttributeValue.N("2022"))
    parametersMovie1.add(AttributeValue.S("My Movie 1"))
    parametersMovie1.add(AttributeValue.S("No Information"))

    val statementRequestMovie1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie1
        }

    // Set data for Movie 2.
    val parametersMovie2 = mutableListOf<AttributeValue>()
    parametersMovie2.add(AttributeValue.N("2022"))
    parametersMovie2.add(AttributeValue.S("My Movie 2"))
    parametersMovie2.add(AttributeValue.S("No Information"))

    val statementRequestMovie2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie2
        }

    // Set data for Movie 3.
    val parametersMovie3 = mutableListOf<AttributeValue>()
    parametersMovie3.add(AttributeValue.N("2022"))
    parametersMovie3.add(AttributeValue.S("My Movie 3"))
    parametersMovie3.add(AttributeValue.S("No Information"))
```

```
    val statementRequestMovie3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestMovie1)
    myBatchStatementList.add(statementRequestMovie2)
    myBatchStatementList.add(statementRequestMovie3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }
    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: " + response.toString())
    println("Added new movies using a batch command.")
}

suspend fun updateTableItemBatchBatch(ddb: DynamoDbClient) {
    val sqlStatement =
        "UPDATE MoviesPartiQBatch SET info = 'directors\":[\"Merian C. Cooper\",
\"Ernest B. Schoedsack' where year=? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))
    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }

    // Update record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec2
        }
```

```
    // Update record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }

    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: $response")
    println("Updated three movies using a batch command.")
    println("Items were updated!")
}

suspend fun deleteItemsBatch(ddb: DynamoDbClient) {
    // Specify three records to delete.
    val sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))

    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }

    // Specify record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
```

```
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec2
        }

    // Specify record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }

    ddb.batchExecuteStatement(batchRequest)
    println("Deleted three movies using a batch command.")
}

suspend fun deleteTablePartiQLBatch(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「BatchExecuteStatement」を参照してください。

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun main() {
    val ddb = DynamoDbClient { region = "us-east-1" }
    val tableName = "MoviesPartiQ"
    val fileName = "../../../resources/sample_files/movies.json"
    println("Creating an Amazon DynamoDB table named MoviesPartiQ with a key named
 id and a sort key named title.")
    createTablePartiQL(ddb, tableName, "year")
    loadDataPartiQL(ddb, fileName)

    println("******* Getting data from the MoviesPartiQ table.")
    getMoviePartiQL(ddb)

    println("******* Putting a record into the MoviesPartiQ table.")
    putRecordPartiQL(ddb)

    println("******* Updating a record.")
    updateTableItemPartiQL(ddb)
```

```
    println("******* Querying the movies released in 2013.")
    queryTablePartiQL(ddb)

    println("******* Deleting the MoviesPartiQ table.")
    deleteTablePartiQL(tableName)
}

suspend fun createTablePartiQL(
    ddb: DynamoDbClient,
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }
```

```
    val response = ddb.createTable(request)
    ddb.waitUntilTableExists {
        // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}

suspend fun loadDataPartiQL(
    ddb: DynamoDbClient,
    fileName: String,
) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
 'info' : ?}"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode
    var t = 0

    while (iter.hasNext()) {
        if (t == 200) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
        parameters.add(AttributeValue.N(year.toString()))
        parameters.add(AttributeValue.S(title))
        parameters.add(AttributeValue.S(info))

        executeStatementPartiQL(ddb, sqlStatement, parameters)
        println("Added Movie $title")
        parameters.clear()
        t++
    }
}
```

```
suspend fun getMoviePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?"
    val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
    parameters.add(AttributeValue.N("2012"))
    parameters.add(AttributeValue.S("The Perks of Being a Wallflower"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun putRecordPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
 'info' : ?}"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2020"))
    parameters.add(AttributeValue.S("My Movie"))
    parameters.add(AttributeValue.S("No Info"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Added new movie.")
}

suspend fun updateTableItemPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian C.
 Cooper\",\"Ernest B. Schoedsack\' where year=? and title=?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    parameters.add(AttributeValue.S("The East"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Item was updated!")
}

// Query the table where the year is 2013.
suspend fun queryTablePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year = ?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun deleteTablePartiQL(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }
```

```
    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun executeStatementPartiQL(
    ddb: DynamoDbClient,
    statementVal: String,
    parametersVal: List<AttributeValue>,
): ExecuteStatementResponse {
    val request =
        ExecuteStatementRequest {
            statement = statementVal
            parameters = parametersVal
        }

    return ddb.executeStatement(request)
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ExecuteStatement」を参照してください。

SDK for Kotlin を使用した Amazon EC2 の例

次のコード例は、Amazon EC2 で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->

        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
 ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeSecurityGroups」を参照してください。

トピック

- 基本
- アクション

基本

基本を学ぶ

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。
- 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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin 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.
 16. Deletes the security group.
 17. Deletes the key pair.
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <keyName> <fileName> <groupName> <groupDesc> <vpcId> <myIpAddress>

        Where:
            keyName - A key pair name (for example, TestKeyPair).
            fileName - A file name where the key information is written to.
            groupName - The name of the security group.
            groupDesc - The description of the security group.
            vpcId - A VPC ID. You can get this value from the AWS Management
 Console.
            myIpAddress - The IP address of your development machine.

"""

    if (args.size != 6) {
        println(usage)
        exitProcess(0)
    }

    val keyName = args[0]
    val fileName = args[1]
    val groupName = args[2]
    val groupDesc = args[3]
    val vpcId = args[4]
    val myIpAddress = args[5]
    var newInstanceId: String? = ""
```

```
    println(DASHES)
    println("Welcome to the Amazon EC2 example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create an RSA key pair and save the private key material as a .pem
file.")
    createKeyPairSc(keyName, fileName)
    println(DASHES)

    println(DASHES)
    println("2. List key pairs.")
    describeEC2KeysSc()
    println(DASHES)

    println(DASHES)
    println("3. Create a security group.")
    val groupId = createEC2SecurityGroupSc(groupName, groupDesc, vpcId, myIpAddress)
    println(DASHES)

    println(DASHES)
    println("4. Display security group info for the newly created security group.")
    describeSecurityGroupsSc(groupId.toString())
    println(DASHES)

    println(DASHES)
    println("5. Get a list of Amazon Linux 2 AMIs and select one with amzn2 in the
name.")
    val instanceId = getParaValuesSc()
    if (instanceId == "") {
        println("The instance Id value isn't valid.")
        exitProcess(0)
    }
    println("The instance Id is $instanceId.")
    println(DASHES)

    println(DASHES)
    println("6. Get more information about an amzn2 image and return the AMI
value.")
    val amiValue = instanceId?.let { describeImageSc(it) }
    if (instanceId == "") {
        println("The instance Id value is invalid.")
        exitProcess(0)
    }
```

```
    println("The AMI value is $amiValue.")
    println(DASHES)

    println(DASHES)
    println("7. Get a list of instance types.")
    val instanceType = getInstanceTypesSc()
    println(DASHES)

    println(DASHES)
    println("8. Create an instance.")
    if (amiValue != null) {
        newInstanceId = runInstanceSc(instanceType, keyName, groupName, amiValue)
        println("The instance Id is $newInstanceId")
    }
    println(DASHES)

    println(DASHES)
    println("9. Display information about the running instance. ")
    var ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("10.  Stop the instance.")
    if (newInstanceId != null) {
        stopInstanceSc(newInstanceId)
    }
    println(DASHES)

    println(DASHES)
    println("11.  Start the instance.")
    if (newInstanceId != null) {
        startInstanceSc(newInstanceId)
    }
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("12. Allocate an Elastic IP address and associate it with the
instance.")
    val allocationId = allocateAddressSc()
```

```
    println("The allocation Id value is $allocationId")
    val associationId = associateAddressSc(newInstanceId, allocationId)
    println("The associate Id value is $associationId")
    println(DASHES)

    println(DASHES)
    println("13. Describe the instance again.")
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("14. Disassociate and release the Elastic IP address.")
    disassociateAddressSc(associationId)
    releaseEC2AddressSc(allocationId)
    println(DASHES)

    println(DASHES)
    println("15. Terminate the instance and use a waiter.")
    if (newInstanceId != null) {
        terminateEC2Sc(newInstanceId)
    }
    println(DASHES)

    println(DASHES)
    println("16. Delete the security group.")
    if (groupId != null) {
        deleteEC2SecGroupSc(groupId)
    }
    println(DASHES)

    println(DASHES)
    println("17. Delete the key pair.")
    deleteKeysSc(keyName)
    println(DASHES)

    println(DASHES)
    println("You successfully completed the Amazon EC2 scenario.")
    println(DASHES)
}

suspend fun deleteKeysSc(keyPair: String) {
    val request =
```

```
        DeleteKeyPairRequest {
            keyName = keyPair
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}

suspend fun deleteEC2SecGroupSc(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted security group with Id $groupIdVal")
    }
}

suspend fun terminateEC2Sc(instanceIdVal: String) {
    val ti =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceIdVal)
        }
    println("Wait for the instance to terminate. This will take a few minutes.")
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.terminateInstances(ti)
        ec2.waitUntilInstanceTerminated {
            // suspend call
            instanceIds = listOf(instanceIdVal)
        }
        println("$instanceIdVal is terminated!")
    }
}

suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
```

```
        println("Successfully released Elastic IP address $allocId")
    }
}

suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}

suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}

suspend fun allocateAddressSc(): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        return allocateResponse.allocationId
    }
}

suspend fun startInstanceSc(instanceId: String) {
    val request =
```

```
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
 minutes.")
        ec2.waitUntilInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}

suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
 minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}

suspend fun describeEC2InstancesSc(newInstanceId: String?): String {
    var pubAddress = ""
    var isRunning = false
    val request =
        DescribeInstancesRequest {
            instanceIds = listOf(newInstanceId.toString())
        }

    while (!isRunning) {
        Ec2Client { region = "us-west-2" }.use { ec2 ->
```

```
            val response = ec2.describeInstances(request)
            val state =
                response.reservations
                    ?.get(0)
                    ?.instances
                    ?.get(0)
                    ?.state
                    ?.name
                    ?. value
            if (state != null) {
                if (state.compareTo("running") == 0) {
                    println("Image id is
 ${response.reservations!!.get(0).instances?.get(0)?.imageId}")
                    println("Instance type is
 ${response.reservations!!.get(0).instances?.get(0)?.instanceType}")
                    println("Instance state is
 ${response.reservations!!.get(0).instances?.get(0)?.state}")
                    pubAddress =
                        response.reservations!!
                            .get(0)
                            .instances
                            ?.get(0)
                            ?.publicIpAddress
                            .toString()
                    println("Instance address is $pubAddress")
                    isRunning = true
                }
            }
        }
    }
    return pubAddress
}

suspend fun runInstanceSc(
    instanceTypeVal: String,
    keyNameVal: String,
    groupNameVal: String,
    amiIdVal: String,
): String {
    val runRequest =
        RunInstancesRequest {
            instanceType = InstanceType.fromValue(instanceTypeVal)
            keyName = keyNameVal
            securityGroups = listOf(groupNameVal)
```

```
            maxCount = 1
            minCount = 1
            imageId = amiIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(runRequest)
        val instanceId = response.instances?.get(0)?.instanceId
        println("Successfully started EC2 Instance $instanceId based on AMI
 $amiIdVal")
        return instanceId.toString()
    }
}

// Get a list of instance types.
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
 ${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
 ${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}

// Display the Description field that corresponds to the instance Id value.
```

```
suspend fun describeImageSc(instanceId: String): String? {
    val imagesRequest =
        DescribeImagesRequest {
            imageIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeImages(imagesRequest)
        println("The description of the first image is
 ${response.images?.get(0)?.description}")
        println("The name of the first image is  ${response.images?.get(0)?.name}")

        // Return the image Id value.
        return response.images?.get(0)?.imageId
    }
}

// Get the Id value of an instance with amzn2 in the name.
suspend fun getParaValuesSc(): String? {
    val parameterRequest =
        GetParametersByPathRequest {
            path = "/aws/service/ami-amazon-linux-latest"
        }

    SsmClient { region = "us-west-2" }.use { ssmClient ->
        val response = ssmClient.getParametersByPath(parameterRequest)
        response.parameters?.forEach { para ->
            println("The name of the para is: ${para.name}")
            println("The type of the para is: ${para.type}")
            println("")
            if (para.name?.let { filterName(it) } == true) {
                return para.value
            }
        }
    }
    return ""
}

fun filterName(name: String): Boolean {
    val parts = name.split("/").toTypedArray()
    val myValue = parts[4]
    return myValue.contains("amzn2")
}
```

```
suspend fun describeSecurityGroupsSc(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeSecurityGroups(request)
        for (group in response.securityGroups!!) {
            println("Found Security Group with id " + group.groupId.toString() + "
 and group VPC " + group.vpcId)
        }
    }
}

suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
```

```
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group $groupNameVal")
        return resp.groupId
    }
}

suspend fun describeEC2KeysSc() {
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
 ${ keyPair.keyFingerprint}")
        }
    }
}

suspend fun createKeyPairSc(
    keyNameVal: String,
    fileNameVal: String,
) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        val content = response.keyMaterial
        if (content != null) {
            File(fileNameVal).writeText(content)
        }
        println("Successfully created key pair named $keyNameVal")
```

```
    }
}
```

- API の詳細については、「AWS SDK for Kotlin 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

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getAllocateAddress(instanceIdVal: String?): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        val allocationIdVal = allocateResponse.allocationId

        val request =
            AssociateAddressRequest {
                instanceId = instanceIdVal
                allocationId = allocationIdVal
            }

        val associateResponse = ec2.associateAddress(request)
        return associateResponse.associationId
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「AllocateAddress」を参照してください。

`AssociateAddress`

次の例は、`AssociateAddress` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「AssociateAddress」を参照してください。

`AuthorizeSecurityGroupIngress`

次の例は、`AuthorizeSecurityGroupIngress` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group $groupNameVal")
```

```
        return resp.groupId
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「AuthorizeSecurityGroupIngress」を参照してください。

`CreateKeyPair`

次の例は、`CreateKeyPair` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createEC2KeyPair(keyNameVal: String) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        println("The key ID is ${response.keyPairId}")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateKeyPair」を参照してください。

`CreateSecurityGroup`

次の例は、`CreateSecurityGroup` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createEC2SecurityGroup(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "0.0.0.0/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }
```

```
        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group $groupNameVal")
        return resp.groupId
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateSecurityGroup」を参照してください。

`DeleteKeyPair`

次の例は、`DeleteKeyPair` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteKeys(keyPair: String?) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteKeyPair」を参照してください。

`DeleteSecurityGroup`

次の例は、`DeleteSecurityGroup` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteEC2SecGroup(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted Security Group with id $groupIdVal")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteSecurityGroup」を参照してください。

`DescribeInstanceTypes`

次の例は、`DescribeInstanceTypes` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Get a list of instance types.
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
 ${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
 ${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeInstanceTypes」を参照してください。

DescribeInstances

次の例は、DescribeInstances を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeEC2Instances() {
    val request =
        DescribeInstancesRequest {
            maxResults = 6
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstances(request)
        response.reservations?.forEach { reservation ->
            reservation.instances?.forEach { instance ->
                println("Instance Id is ${instance.instanceId}")
                println("Image id is ${instance.imageId}")
                println("Instance type is ${instance.instanceType}")
                println("Instance state name is ${instance.state?.name}")
                println("monitoring information is ${instance.monitoring?.state}")
            }
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeInstances」を参照してください。

DescribeKeyPairs

次の例は、DescribeKeyPairs を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeEC2Keys() {
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
 ${ keyPair.keyFingerprint}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeKeyPairs」を参照してください。

`DescribeSecurityGroups`

次の例は、`DescribeSecurityGroups` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }
```

```
    Ec2Client { region = "us-west-2" }.use { ec2 ->

        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
 ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeSecurityGroups」を参照してください。

`DisassociateAddress`

次の例は、`DisassociateAddress` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DisassociateAddress」を参照してください。

`ReleaseAddress`

次の例は、`ReleaseAddress` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ReleaseAddress」を参照してください。

`RunInstances`

次の例は、`RunInstances` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createEC2Instance(
    name: String,
    amiId: String,
): String? {
    val request =
        RunInstancesRequest {
            imageId = amiId
            instanceType = InstanceType.T1Micro
            maxCount = 1
            minCount = 1
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(request)
        val instanceId = response.instances?.get(0)?.instanceId
        val tag =
            Tag {
                key = "Name"
                value = name
            }

        val requestTags =
            CreateTagsRequest {
                resources = listOf(instanceId.toString())
                tags = listOf(tag)
            }
        ec2.createTags(requestTags)
        println("Successfully started EC2 Instance $instanceId based on AMI $amiId")
        return instanceId
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「RunInstances」を参照してください。

`StartInstances`

次の例は、`StartInstances` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
 minutes.")
        ec2.waitUntilInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「StartInstances」を参照してください。

`StopInstances`

次の例は、`StopInstances` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
 minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「StopInstances」を参照してください。

`TerminateInstances`

次の例は、`TerminateInstances` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun terminateEC2(instanceID: String) {
    val request =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceID)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
```

```
        val response = ec2.terminateInstances(request)
        response.terminatingInstances?.forEach { instance ->
            println("The ID of the terminated instance is ${instance.instanceId}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「TerminateInstances」を参照してください。

SDK for Kotlin を使用した Amazon ECR の例

次のコード例は、Amazon ECR で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

Hello Amazon ECR

次のコード例は、Amazon ECR の使用を開始する方法を示しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import aws.sdk.kotlin.services.ecr.EcrClient
```

```
import aws.sdk.kotlin.services.ecr.model.ListImagesRequest
import kotlin.system.exitProcess

suspend fun main(args: Array<String>) {
    val usage = """
            Usage: <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.

    """.trimIndent()

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    val repoName = args[0]
    listImageTags(repoName)
}

suspend fun listImageTags(repoName: String?) {
    val listImages =
        ListImagesRequest {
            repositoryName = repoName
        }

    EcrClient { region = "us-east-1" }.use { ecrClient ->
        val imageResponse = ecrClient.listImages(listImages)
        imageResponse.imageIds?.forEach { imageId ->
            println("Image tag: ${imageId.imageTag}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「listImages」を参照してください。

トピック

- 基本
- アクション

基本

基本を学ぶ

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 ECR リポジトリを作成します。
- リポジトリポリシーを設定します。
- リポジトリ URI を取得します。
- Amazon ECR 認可トークンを取得します。
- Amazon ECR リポジトリのライフサイクルポリシーを設定します。
- Docker イメージを Amazon ECR リポジトリにプッシュします。
- Amazon ECR リポジトリにイメージが存在しているか確認します。
- アカウントの Amazon ECR リポジトリをリストし、その詳細を取得します。
- Amazon ECR リポジトリを削除します。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

Amazon ECR 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 java.util.Scanner

/**
 * Before running this Kotlin code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 *
 * This code example requires an IAM Role that has permissions to interact with the
 Amazon ECR service.
```

```
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This code example requires a local docker image named echo-text. Without a local
 image,
 * this program will not successfully run. For more information including how to
 create the local
 * image, see:
 *
 * /scenarios/basics/ecr/README
 *
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage: <iamRoleARN> <accountId>

        Where:
           iamRoleARN - The IAM role ARN that has the necessary permissions to
 access and manage the Amazon ECR repository.
           accountId - Your AWS account number.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        return
    }

    var iamRole = args[0]
    var localImageName: String
    var accountId = args[1]
    val ecrActions = ECRActions()
    val scanner = Scanner(System.`in`)

    println(
        """
        The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
 container registry
```

```
        service provided by AWS. It allows developers and organizations to securely
        store, manage, and deploy Docker container images.
        ECR provides a simple and scalable way to manage container images throughout
their lifecycle,
        from building and testing to production deployment.

        The `EcrClient` service client that is part of the AWS SDK for Kotlin
provides a set of methods to
        programmatically interact with the Amazon ECR service. This allows
developers to
        automate the storage, retrieval, and management of container images as part
of their application
        deployment pipelines. With ECR, teams can focus on building and deploying
their
        applications without having to worry about the underlying infrastructure
required to
        host and manage a container registry.

        This scenario walks you through how to perform key operations for this
service.
        Let's get started...

         You have two choices:
            1 - Run the entire program.
            2 - Delete an existing Amazon ECR repository named echo-text (created
from a previous execution of
            this program that did not complete).

        """.trimIndent(),
    )

    while (true) {
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("1", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else if (input.trim { it <= ' ' }.equals("2", ignoreCase = true)) {
            val repoName = "echo-text"
            ecrActions.deleteECRRepository(repoName)
            return
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
```

```
        }
    }

    waitForInputToContinue(scanner)
    println(DASHES)
    println(
        """
        1. Create an ECR repository.

        The first task is to ensure we have a local Docker image named echo-text.
        If this image exists, then an Amazon ECR repository is created.

        An ECR repository is a private Docker container repository provided
        by Amazon Web Services (AWS). It is a managed service that makes it easy
        to store, manage, and deploy Docker container images.

        """.trimIndent(),
    )

    // Ensure that a local docker image named echo-text exists.
    val doesExist = ecrActions.listLocalImages()
    val repoName: String
    if (!doesExist) {
        println("The local image named echo-text does not exist")
        return
    } else {
        localImageName = "echo-text"
        repoName = "echo-text"
    }

    val repoArn = ecrActions.createECRRepository(repoName).toString()
    println("The ARN of the ECR repository is $repoArn")
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        2. Set an ECR repository policy.

        Setting an ECR repository policy using the `setRepositoryPolicy` function is
crucial for maintaining
        the security and integrity of your container images. The repository policy
allows you to
```

```
        define specific rules and restrictions for accessing and managing the images
stored within your ECR
        repository.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    ecrActions.setRepoPolicy(repoName, iamRole)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val policyText = ecrActions.getRepoPolicy(repoName)
    println("Policy Text:")
    println(policyText)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is responsible
for obtaining a
        valid authorization token, which is required to authenticate your requests
to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
        ECR repository, such as pushing, pulling, or managing your Docker images.


        """.trimIndent(),
```

```
    )
    waitForInputToContinue(scanner)
    ecrActions.getAuthToken()
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        5. Get the ECR Repository URI.

        The URI  of an Amazon ECR repository is important. When you want to deploy a
container image to
        a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
        or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
        which includes the ECR repository URI. This allows the container runtime to
pull the
        correct container image from the ECR repository.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val repositoryURI: String? = ecrActions.getRepositoryURI(repoName)
    println("The repository URI is $repositoryURI")
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        6. Set an ECR Lifecycle Policy.

        An ECR Lifecycle Policy is used to manage the lifecycle of Docker images
stored in your ECR repositories.
        These policies allow you to automatically remove old or unused Docker images
from your repositories,
        freeing up storage space and reducing costs.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val pol = ecrActions.setLifeCyclePolicy(repoName)
    println(pol)
    waitForInputToContinue(scanner)
```

```
    println(DASHES)
    println(
        """
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
repository.
        It sets up the Docker client by connecting to the local Docker host using
the default port.
        It then retrieves the authorization token for the ECR repository by making a
call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
which is used to authenticate
        the Docker client when pushing the image. Finally, the method tags the
Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR
repository using the Docker client.
        If the push operation is successful, the method prints a message indicating
that the image was pushed to ECR.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    ecrActions.pushDockerImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
    println("8. Verify if the image is in the ECR Repository.")
    waitForInputToContinue(scanner)
    ecrActions.verifyImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
    println("9. As an optional step, you can interact with the image in Amazon ECR
by using the CLI.")
    println("Would you like to view instructions on how to use the CLI to run the
image? (y/n)")
    val ans = scanner.nextLine().trim()
    if (ans.equals("y", true)) {
        val instructions = """
```

```
        1. Authenticate with ECR - Before you can pull the image from Amazon ECR,
 you need to authenticate with the registry. You can do this using the AWS CLI:

            aws ecr get-login-password --region us-east-1 | docker login --username
 AWS --password-stdin $accountId.dkr.ecr.us-east-1.amazonaws.com

        2. Describe the image using this command:

           aws ecr describe-images --repository-name $repoName --image-ids imageTag=
$localImageName

        3. Run the Docker container and view the output using this command:

           docker run --rm $accountId.dkr.ecr.us-east-1.amazonaws.com/$repoName:
$localImageName
        """
        println(instructions)
    }
    waitForInputToContinue(scanner)

    println(DASHES)
    println("10. Delete the ECR Repository.")
    println(
        """
        If the repository isn't empty, you must either delete the contents of the
 repository
        or use the force option (used in this scenario) to delete the repository and
 have Amazon ECR delete all of its contents
        on your behalf.

        """.trimIndent(),
    )
    println("Would you like to delete the Amazon ECR Repository? (y/n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
        println("You selected to delete the AWS ECR resources.")
        waitForInputToContinue(scanner)
        ecrActions.deleteECRRepository(repoName)
    }

    println(DASHES)
    println("This concludes the Amazon ECR SDK scenario")
    println(DASHES)
}
```

```
private fun waitForInputToContinue(scanner: Scanner) {
    while (true) {
        println("")
        println("Enter 'c' followed by <ENTER> to continue:")
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
        }
    }
}
```

Amazon ECR SDK メソッドのラッパークラス。

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.CreateRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DeleteRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DescribeImagesRequest
import aws.sdk.kotlin.services.ecr.model.DescribeRepositoriesRequest
import aws.sdk.kotlin.services.ecr.model.EcrException
import aws.sdk.kotlin.services.ecr.model.GetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.ImageIdentifier
import aws.sdk.kotlin.services.ecr.model.RepositoryAlreadyExistsException
import aws.sdk.kotlin.services.ecr.model.SetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.StartLifecyclePolicyPreviewRequest
import com.github.dockerjava.api.DockerClient
import com.github.dockerjava.api.command.DockerCmdExecFactory
import com.github.dockerjava.api.model.AuthConfig
import com.github.dockerjava.core.DockerClientBuilder
import com.github.dockerjava.netty.NettyDockerCmdExecFactory
import java.io.IOException
import java.util.Base64

class ECRActions {
    private var dockerClient: DockerClient? = null

    private fun getDockerClient(): DockerClient? {
```

```
        val osName = System.getProperty("os.name")
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            val dockerHost = "tcp://localhost:2375" // Use the Docker Desktop
default port.
            val dockerCmdExecFactory: DockerCmdExecFactory =

NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000)
            dockerClient =
DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactory).
        } else {
            dockerClient = DockerClientBuilder.getInstance().build()
        }
        return dockerClient
    }


    /**
     * Sets the lifecycle policy for the specified repository.
     *
     * @param repoName the name of the repository for which to set the lifecycle
policy.
     */
    suspend fun setLifeCyclePolicy(repoName: String): String? {
        val polText =
            """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
                     },
                     "action": {
                         "type": "expire"
                     }
                 }
             ]
             }
```

```
            """.trimIndent()
        val lifecyclePolicyPreviewRequest =
            StartLifecyclePolicyPreviewRequest {
                lifecyclePolicyText = polText
                repositoryName = repoName
            }

        // Execute the request asynchronously.
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest)
            return response.lifecyclePolicyText
        }
    }


    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }
```

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }


    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }


    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
```

```
     * @param iamRole the IAM role to be granted access to the repository.
     */
    suspend fun setRepoPolicy(
        repoName: String?,
        iamRole: String?,
    ) {
        val policyDocumentTemplate =
            """
             {
              "Version" : "2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }


    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an empty
string if the operation failed.
     * @throws RepositoryAlreadyExistsException if the repository exists.
```

```
     * @throws EcrException         if an error occurs while creating the
repository.
     */
    suspend fun createECRRepository(repoName: String?): String? {
        val request =
            CreateRepositoryRequest {
                repositoryName = repoName
            }

        return try {
            EcrClient { region = "us-east-1" }.use { ecrClient ->
                val response = ecrClient.createRepository(request)
                response.repository?.repositoryArn
            }
        } catch (e: RepositoryAlreadyExistsException) {
            println("Repository already exists: $repoName")
            repoName?.let { getRepoARN(it) }
        } catch (e: EcrException) {
            println("An error occurred: ${e.message}")
            null
        }
    }

    suspend fun getRepoARN(repoName: String): String? {
        // Fetch the existing repository's ARN.
        val describeRequest =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeResponse = ecrClient.describeRepositories(describeRequest)
            return describeResponse.repositories?.get(0)?.repositoryArn
        }
    }

    fun listLocalImages(): Boolean = try {
        val images = getDockerClient()?.listImagesCmd()?.exec()
        images?.any { image ->
            image.repoTags?.any { tag -> tag.startsWith("echo-text") } ?: false
        } ?: false
    } catch (ex: Exception) {
        println("ERROR: ${ex.message}")
        false
    }
```

```
   /**
    * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
    *
    * @param repoName the name of the ECR repository to push the image to.
    * @param imageName the name of the Docker image.
    */
   suspend fun pushDockerImage(
       repoName: String,
       imageName: String,
   ) {
       println("Pushing $imageName to $repoName will take a few seconds")
       val authConfig = getAuthConfig(repoName)

       EcrClient { region = "us-east-1" }.use { ecrClient ->
           val desRequest =
               DescribeRepositoriesRequest {
                   repositoryNames = listOf(repoName)
               }

           val describeRepoResponse = ecrClient.describeRepositories(desRequest)
           val repoData =
               describeRepoResponse.repositories?.firstOrNull { it.repositoryName
== repoName }
                   ?: throw RuntimeException("Repository not found: $repoName")

           val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
           if (tagImageCmd != null) {
               tagImageCmd.exec()
           }
           val pushImageCmd =
               repoData.repositoryUri?.let {
                   dockerClient?.pushImageCmd(it)
                       // ?.withTag("latest")
                       ?.withAuthConfig(authConfig)
               }

           try {
               if (pushImageCmd != null) {
                   pushImageCmd.start().awaitCompletion()
               }
```

```
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }


    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
 be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
 be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

```
    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }

    // Return an AuthConfig.
    private suspend fun getAuthConfig(repoName: String): AuthConfig {
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            val decodedToken = String(Base64.getDecoder().decode(token))
            val password = decodedToken.substring(4)

            val request =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val descrRepoResponse = ecrClient.describeRepositories(request)
            val repoData = descrRepoResponse.repositories?.firstOrNull
{ it.repositoryName == repoName }
```

```
            val registryURL: String = repoData?.repositoryUri?.split("/")?.get(0) ?:
 ""

            return AuthConfig()
                .withUsername("AWS")
                .withPassword(password)
                .withRegistryAddress(registryURL)
        }
    }
}
```

- API の詳細については、『AWS SDK for Kotlin API リファレンス』の以下のトピックを参照してください。
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

アクション

`CreateRepository`

次の例は、`CreateRepository` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an empty
string if the operation failed.
     * @throws RepositoryAlreadyExistsException if the repository exists.
     * @throws EcrException         if an error occurs while creating the
repository.
     */
    suspend fun createECRRepository(repoName: String?): String? {
        val request =
            CreateRepositoryRequest {
                repositoryName = repoName
            }

        return try {
            EcrClient { region = "us-east-1" }.use { ecrClient ->
                val response = ecrClient.createRepository(request)
                response.repository?.repositoryArn
            }
        } catch (e: RepositoryAlreadyExistsException) {
            println("Repository already exists: $repoName")
            repoName?.let { getRepoARN(it) }
        } catch (e: EcrException) {
            println("An error occurred: ${e.message}")
            null
        }
    }
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateRepository」を参照してください。

`DeleteRepository`

次の例は、DeleteRepository を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteRepository」を参照してください。

`DescribeImages`

次の例は、`DescribeImages` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
 be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
 be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
```

```
            }
        }
    }
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeImages」を参照してください。

`DescribeRepositories`

次の例は、`DescribeRepositories` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
```

```
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeRepositories」を参照してください。

`GetAuthorizationToken`

次の例は、`GetAuthorizationToken` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetAuthorizationToken」を参照してください。

`GetRepositoryPolicy`

次の例は、`GetRepositoryPolicy` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetRepositoryPolicy」を参照してください。

PushImageCmd

次の例は、PushImageCmd を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    suspend fun pushDockerImage(
        repoName: String,
        imageName: String,
    ) {
        println("Pushing $imageName to $repoName will take a few seconds")
        val authConfig = getAuthConfig(repoName)

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val desRequest =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val describeRepoResponse = ecrClient.describeRepositories(desRequest)
            val repoData =
                describeRepoResponse.repositories?.firstOrNull { it.repositoryName
== repoName }
                    ?: throw RuntimeException("Repository not found: $repoName")

            val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
            if (tagImageCmd != null) {
                tagImageCmd.exec()
```

```
            }
            val pushImageCmd =
                repoData.repositoryUri?.let {
                    dockerClient?.pushImageCmd(it)
                        // ?.withTag("latest")
                        ?.withAuthConfig(authConfig)
                }

            try {
                if (pushImageCmd != null) {
                    pushImageCmd.start().awaitCompletion()
                }
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「PushImageCmd」を参照してください。

`SetRepositoryPolicy`

次の例は、`SetRepositoryPolicy` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Sets the repository policy for the specified ECR repository.
 *
 * @param repoName the name of the ECR repository.
 * @param iamRole the IAM role to be granted access to the repository.
```

```
     */
    suspend fun setRepoPolicy(
        repoName: String?,
        iamRole: String?,
    ) {
        val policyDocumentTemplate =
            """
             {
              "Version" : "2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「SetRepositoryPolicy」を参照してください。

`StartLifecyclePolicyPreview`

次の例は、`StartLifecyclePolicyPreview` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
 be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
 be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
```

```
            }
        }
    }
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「StartLifecyclePolicyPreview」を参照してください。

SDK for Kotlin を使用した OpenSearch Service の例

次のコード例は、OpenSearch Service で AWS SDK for Kotlin 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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createNewDomain(domainNameVal: String?) {
```

```
    val clusterConfigOb =
        ClusterConfig {
            dedicatedMasterEnabled = true
            dedicatedMasterCount = 3
            dedicatedMasterType =
 OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceType =
 OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceCount = 5
        }

    val ebsOptionsOb =
        EbsOptions {
            ebsEnabled = true
            volumeSize = 10
            volumeType = VolumeType.Gp2
        }

    val encryptionOptionsOb =
        NodeToNodeEncryptionOptions {
            enabled = true
        }

    val request =
        CreateDomainRequest {
            domainName = domainNameVal
            engineVersion = "OpenSearch_1.0"
            clusterConfig = clusterConfigOb
            ebsOptions = ebsOptionsOb
            nodeToNodeEncryptionOptions = encryptionOptionsOb
        }

    println("Sending domain creation request...")
    OpenSearchClient { region = "us-east-1" }.use { searchClient ->
        val createResponse = searchClient.createDomain(request)
        println("Domain status is ${createResponse.domainStatus}")
        println("Domain Id is ${createResponse.domainStatus?.domainId}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CreateDomain」を参照してください。

`DeleteDomain`

次の例は、`DeleteDomain` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteSpecificDomain(domainNameVal: String) {
    val request =
        DeleteDomainRequest {
            domainName = domainNameVal
        }
    OpenSearchClient { region = "us-east-1" }.use { searchClient ->
        searchClient.deleteDomain(request)
        println("$domainNameVal was successfully deleted.")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteDomain」を参照してください。

`ListDomainNames`

次の例は、`ListDomainNames` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listAllDomains() {
```

```
    OpenSearchClient { region = "us-east-1" }.use { searchClient ->
        val response: ListDomainNamesResponse =
 searchClient.listDomainNames(ListDomainNamesRequest {})
        response.domainNames?.forEach { domain ->
            println("Domain name is " + domain.domainName)
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ListDomainNames」を参照してください。

`UpdateDomainConfig`

次の例は、`UpdateDomainConfig` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun updateSpecificDomain(domainNameVal: String?) {
    val clusterConfigOb =
        ClusterConfig {
            instanceCount = 3
        }

    val request =
        UpdateDomainConfigRequest {
            domainName = domainNameVal
            clusterConfig = clusterConfigOb
        }

    println("Sending domain update request...")
    OpenSearchClient { region = "us-east-1" }.use { searchClient ->
        val updateResponse = searchClient.updateDomainConfig(request)
        println("Domain update response from Amazon OpenSearch Service:")
```

```
        println(updateResponse.toString())
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「UpdateDomainConfig」を参照してください。

SDK for Kotlin を使用した EventBridge の例

次のコード例は、EventBridge で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

Hello EventBridge

次のコード例は、EventBridge の使用を開始する方法を示しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import aws.sdk.kotlin.services.eventbridge.EventBridgeClient
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesRequest
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesResponse
```

```
suspend fun main() {
    listBusesHello()
}

suspend fun listBusesHello() {
    val request =
        ListEventBusesRequest {
            limit = 10
        }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val response: ListEventBusesResponse = eventBrClient.listEventBuses(request)
        response.eventBuses?.forEach { bus ->
            println("The name of the event bus is ${bus.name}")
            println("The ARN of the event bus is ${bus.arn}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ListEventBuses」を参照してください。

トピック

- 基本
- アクション

基本

基本を学ぶ

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/*
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin example performs the following tasks with Amazon EventBridge:

 1. Creates an AWS Identity and Access Management (IAM) role to use with Amazon
 EventBridge.
 2. Creates an 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 S3 bucket.
 11. Disables a specific rule.
 12. Checks and prints 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 S3 bucket.
 16. Updates the rule to a custom rule pattern.
 17. Sends an event to trigger the rule.
 18. Cleans up resources.
*/
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
```

```
suspend fun main(args: Array<String>) {
    val 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.
    """
    val polJSON =
        "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
            "\"Effect\": \"Allow\"," +
            "\"Principal\": {" +
            "\"Service\": \"events.amazonaws.com\"" +
            "}," +
            "\"Action\": \"sts:AssumeRole\"" +
            "}]" +
            "}"

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val sc = Scanner(System.`in`)
    val roleName = args[0]
    val bucketName = args[1]
    val topicName = args[2]
    val eventRuleName = args[3]

    println(DASHES)
    println("Welcome to the Amazon EventBridge example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS Identity and Access Management (IAM) role to use with
 Amazon EventBridge.")
    val roleArn = createIAMRole(roleName, polJSON)
```

```
    println(DASHES)

    println(DASHES)
    println("2. Create an S3 bucket with EventBridge events enabled.")
    if (checkBucket(bucketName)) {
        println("$bucketName already exists. Ending this scenario.")
        exitProcess(1)
    }

    createBucket(bucketName)
    delay(3000)
    setBucketNotification(bucketName)
    println(DASHES)

    println(DASHES)
    println("3. Create a rule that triggers when an object is uploaded to Amazon
S3.")
    delay(10000)
    addEventRule(roleArn, bucketName, eventRuleName)
    println(DASHES)

    println(DASHES)
    println("4. List rules on the event bus.")
    listRules()
    println(DASHES)

    println(DASHES)
    println("5. Create a new SNS topic for testing and let the user subscribe to the
topic.")
    val topicArn = createSnsTopic(topicName)
    println(DASHES)

    println(DASHES)
    println("6. Add a target to the rule that sends an email to the specified
topic.")
    println("Enter your email to subscribe to the Amazon SNS topic:")
    val email = sc.nextLine()
    subEmail(topicArn, email)
    println("Use the link in the email you received to confirm your subscription.
Then press Enter to continue.")
    sc.nextLine()
    println(DASHES)

    println(DASHES)
```

```
    println("7. Create an EventBridge event that sends an email when an Amazon S3
object is created.")
    addSnsEventRule(eventRuleName, topicArn, topicName, eventRuleName, bucketName)
    println(DASHES)

    println(DASHES)
    println("8. List targets.")
    listTargets(eventRuleName)
    println(DASHES)

    println(DASHES)
    println(" 9. List the rules for the same target.")
    listTargetRules(topicArn)
    println(DASHES)

    println(DASHES)
    println("10. Trigger the rule by uploading a file to the S3 bucket.")
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
    println("11. Disable a specific rule.")
    changeRuleState(eventRuleName, false)
    println(DASHES)

    println(DASHES)
    println("12. Check and print the state of the rule.")
    checkRule(eventRuleName)
    println(DASHES)

    println(DASHES)
    println("13. Add a transform to the rule to change the text of the email.")
    updateSnsEventRule(topicArn, eventRuleName)
    println(DASHES)

    println(DASHES)
    println("14. Enable a specific rule.")
    changeRuleState(eventRuleName, true)
    println(DASHES)

    println(DASHES)
    println("15. Trigger the updated rule by uploading a file to the S3 bucket.")
```

```
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
    println("16. Update the rule to a custom rule pattern.")
    updateToCustomRule(eventRuleName)
    println("Updated event rule $eventRuleName to use a custom pattern.")
    updateCustomRuleTargetWithTransform(topicArn, eventRuleName)
    println("Updated event target $topicArn.")
    println(DASHES)

    println(DASHES)
    println("17. Send an event to trigger the rule. This will trigger a subscription
 email.")
    triggerCustomRule(email)
    println("Events have been sent. Press Enter to continue.")
    sc.nextLine()
    println(DASHES)

    println(DASHES)
    println("18. Clean up resources.")
    println("Do you want to clean up resources (y/n)")
    val ans = sc.nextLine()
    if (ans.compareTo("y") == 0) {
        cleanupResources(topicArn, eventRuleName, bucketName, roleName)
    } else {
        println("The resources will not be cleaned up. ")
    }
    println(DASHES)

    println(DASHES)
    println("The Amazon EventBridge example scenario has successfully completed.")
    println(DASHES)
}

suspend fun cleanupResources(
    topicArn: String?,
    eventRuleName: String?,
    bucketName: String?,
    roleName: String?,
) {
    println("Removing all targets from the event rule.")
```

```
    deleteTargetsFromRule(eventRuleName)
    deleteRuleByName(eventRuleName)
    deleteSNSTopic(topicArn)
    deleteS3Bucket(bucketName)
    deleteRole(roleName)
}

suspend fun deleteRole(roleNameVal: String?) {
    val policyArnVal = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
    val policyRequest =
        DetachRolePolicyRequest {
            policyArn = policyArnVal
            roleName = roleNameVal
        }
    IamClient { region = "us-east-1" }.use { iam ->
        iam.detachRolePolicy(policyRequest)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")

        // Delete the role.
        val roleRequest =
            DeleteRoleRequest {
                roleName = roleNameVal
            }

        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteS3Bucket(bucketName: String?) {
    // Remove all the objects from the S3 bucket.
    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }
    S3Client { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val myObjects = res.contents
        val toDelete = mutableListOf<ObjectIdentifier>()

        if (myObjects != null) {
            for (myValue in myObjects) {
                toDelete.add(
                    ObjectIdentifier {
```

```
                        key = myValue.key
                    },
                )
            }
        }

        val delOb =
            Delete {
                objects = toDelete
            }

        val dor =
            DeleteObjectsRequest {
                bucket = bucketName
                delete = delOb
            }
        s3Client.deleteObjects(dor)

        // Delete the S3 bucket.
        val deleteBucketRequest =
            DeleteBucketRequest {
                bucket = bucketName
            }
        s3Client.deleteBucket(deleteBucketRequest)
        println("You have deleted the bucket and the objects")
    }
}

// Delete the SNS topic.
suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println(" $topicArnVal was deleted.")
    }
}

suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
```

```
            name = ruleName
        }
    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}

suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest =
                    RemoveTargetsRequest {
                        rule = eventRuleName
                        ids = listOf(myTarget.id.toString())
                    }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
}

suspend fun triggerCustomRule(email: String) {
    val json =
        "{" +
            "\"UserEmail\": \"" + email + "\"," +
            "\"Message\": \"This event was generated by example code.\"" +
            "\"UtcTime\": \"Now.\"" +
            "}"

    val entry =
        PutEventsRequestEntry {
```

```
            source = "ExampleSource"
            detail = json
            detailType = "ExampleType"
        }

    val eventsRequest =
        PutEventsRequest {
            this.entries = listOf(entry)
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putEvents(eventsRequest)
    }
}

suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\""
        }

    val target =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformerOb
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(target)
            eventBusName = null
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}
```

```
suspend fun updateToCustomRule(ruleName: String?) {
    val customEventsPattern =
        "{" +
            "\"source\": [\"ExampleSource\"]," +
            "\"detail-type\": [\"ExampleType\"]" +
            "}"
    val request =
        PutRuleRequest {
            name = ruleName
            description = "Custom test rule"
            eventPattern = customEventsPattern
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putRule(request)
    }
}

// Update an Amazon S3 object created rule with a transform on the target.
suspend fun updateSnsEventRule(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()
    val myMap = mutableMapOf<String, String>()
    myMap["bucket"] = "$.detail.bucket.name"
    myMap["time"] = "$.time"

    val inputTransOb =
        InputTransformer {
            inputTemplate = "\"Notification: an object was uploaded to bucket
 <bucket> at <time>.\""
            inputPathsMap = myMap
        }
    val targetOb =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransOb
        }

    val targetsRequest =
        PutTargetsRequest {
```

```
            rule = ruleName
            targets = listOf(targetOb)
            eventBusName = null
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}

suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
```

```
    }
}

// Create and upload a file to an S3 bucket to trigger an event.
@Throws(IOException::class)
suspend fun uploadTextFiletoS3(bucketName: String?) {
    val fileSuffix = SimpleDateFormat("yyyyMMddHHmmss").format(Date())
    val fileName = "TextFile$fileSuffix.txt"
    val myFile = File(fileName)
    val fw = FileWriter(myFile.absoluteFile)
    val bw = BufferedWriter(fw)
    bw.write("This is a sample file for testing uploads.")
    bw.close()

    val putOb =
        PutObjectRequest {
            bucket = bucketName
            key = fileName
            body = myFile.asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.putObject(putOb)
    }
}

suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}

suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
```

```
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}

// Add a rule that triggers an SNS target when a file is uploaded to an S3 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
            id = targetID
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request =
        PutTargetsRequest {
            eventBusName = null
            targets = targetsOb
            rule = ruleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target $topicName
 for bucket $bucketName.")
    }
}

suspend fun subEmail(
    topicArnVal: String?,
```

```
    email: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" Subscription ARN: ${result.subscriptionArn}")
    }
}

suspend fun createSnsTopic(topicName: String): String? {
    val topicPolicy =
        "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
            "\"Sid\": \"EventBridgePublishTopic\"," +
            "\"Effect\": \"Allow\"," +
            "\"Principal\": {" +
            "\"Service\": \"events.amazonaws.com\"" +
            "}," +
            "\"Resource\": \"*\"," +
            "\"Action\": \"sns:Publish\"" +
            "}]" +
            "}"

    val topicAttributes = mutableMapOf<String, String>()
    topicAttributes["Policy"] = topicPolicy

    val topicRequest =
        CreateTopicRequest {
            name = topicName
            attributes = topicAttributes
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        println("Added topic $topicName for email subscriptions.")
        return response.topicArn
```

```
    }
}

suspend fun listRules() {
    val rulesRequest =
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}

// Create a new event rule that triggers when an Amazon S3 object is created in a
 bucket.
suspend fun addEventRule(
    roleArnVal: String?,
    bucketName: String,
    eventRuleName: String?,
) {
    val pattern = """{
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
        "bucket": {
            "name": ["$bucketName"]
            }
        }
    }"""

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }
```

```
    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}

// Set the Amazon S3 bucket notification configuration.
suspend fun setBucketNotification(bucketName: String) {
    val eventBridgeConfig =
        EventBridgeConfiguration {
        }

    val configuration =
        NotificationConfiguration {
            eventBridgeConfiguration = eventBridgeConfig
        }

    val configurationRequest =
        PutBucketNotificationConfigurationRequest {
            bucket = bucketName
            notificationConfiguration = configuration
            skipDestinationValidation = true
        }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.putBucketNotificationConfiguration(configurationRequest)
        println("Added bucket $bucketName with EventBridge events enabled.")
    }
}

// Create an S3 bucket using a waiter.
suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        s3.waitUntilBucketExists {
            bucket = bucketName
        }
        println("$bucketName is ready")
    }
```

```
}

suspend fun checkBucket(bucketName: String?): Boolean {
    try {
        // Determine if the S3 bucket exists.
        val headBucketRequest =
            HeadBucketRequest {
                bucket = bucketName
            }

        S3Client { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            return true
        }
    } catch (e: S3Exception) {
        System.err.println(e.message)
    }
    return false
}

suspend fun createIAMRole(
    rolenameVal: String?,
    polJSON: String?,
): String? {
    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = polJSON
            description = "Created using the AWS SDK for Kotlin"
        }

    val rolePolicyRequest =
        AttachRolePolicyRequest {
            roleName = rolenameVal
            policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
        }

    IamClient { region = "us-east-1" }.use { iam ->
        val response = iam.createRole(request)
        iam.attachRolePolicy(rolePolicyRequest)
        return response.role?.arn
    }
}
```

- API の詳細については、「AWS SDK for Kotlin 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

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.deleteRule(ruleRequest)
```

```
        println("Successfully deleted the rule")
    }
}
```

- API の詳細については、AWS SDK for Rust API リファレンスの「DeleteRule」を参照してください。

`DescribeRule`

次の例は、`DescribeRule` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeRule」を参照してください。

`DisableRule`

次の例は、`DisableRule` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DisableRule」を参照してください。

`EnableRule`

次の例は、`EnableRule` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「EnableRule」を参照してください。

`ListRuleNamesByTarget`

次の例は、`ListRuleNamesByTarget` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ListRuleNamesByTarget」を参照してください。

`ListRules`

次の例は、`ListRules` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listRules() {
    val rulesRequest =
```

```
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ListRules」を参照してください。

`ListTargetsByRule`

次の例は、`ListTargetsByRule` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
```

```
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ListTargetsByRule」を参照してください。

`PutEvents`

次の例は、`PutEvents` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun triggerCustomRule(email: String) {
    val json =
        "{" +
            "\"UserEmail\": \"" + email + "\"," +
            "\"Message\": \"This event was generated by example code.\"" +
            "\"UtcTime\": \"Now.\"" +
            "}"

    val entry =
        PutEventsRequestEntry {
            source = "ExampleSource"
            detail = json
            detailType = "ExampleType"
        }

    val eventsRequest =
        PutEventsRequest {
            this.entries = listOf(entry)
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putEvents(eventsRequest)
```

```
    }
}
```

- API の詳細については、AWS SDK for Rust API リファレンスの「PutEvents」を参照してください。

`PutRule`

次の例は、`PutRule` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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。

```
suspend fun createScRule(
    ruleName: String?,
    cronExpression: String?,
) {
    val ruleRequest =
        PutRuleRequest {
            name = ruleName
            eventBusName = "default"
            scheduleExpression = cronExpression
            state = RuleState.Enabled
            description = "A test rule that runs on a schedule created by the Kotlin
 API"
        }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

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.
suspend fun addEventRule(
    roleArnVal: String?,
    bucketName: String,
    eventRuleName: String?,
) {
    val pattern = """{
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
        "bucket": {
            "name": ["$bucketName"]
            }
        }
    }"""

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

- API の詳細については、AWS SDK for Rust API リファレンスの「PutRule」を参照してください。

`PutTargets`

次の例は、`PutTargets` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Add a rule that triggers an SNS target when a file is uploaded to an S3 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
            id = targetID
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request =
        PutTargetsRequest {
            eventBusName = null
            targets = targetsOb
            rule = ruleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(request)
        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。

```
suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\""
        }

    val target =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformerOb
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(target)
            eventBusName = null
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「PutTargets」を参照してください。

`RemoveTargets`

次の例は、RemoveTargets を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest =
                    RemoveTargetsRequest {
                        rule = eventRuleName
                        ids = listOf(myTarget.id.toString())
                    }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「RemoveTargets」を参照してください。

AWS Glue SDK for Kotlin を使用した の例

次のコード例は、 で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

トピック

- 基本
- アクション

基本

基本を学ぶ

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName> <scriptLocation>
 <locationUri>

        Where:
            iam - The Amazon Resource Name (ARN) of the AWS Identity and Access
 Management (IAM) role that has AWS Glue and Amazon Simple Storage Service (Amazon
 S3) permissions.
            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 (for example,
 cron(15 12 * * ? *).
            dbName - The database name.
            crawlerName - The name of the crawler.
            jobName - The name you assign to this job definition.
            scriptLocation - Specifies the Amazon S3 path to a script that runs a
 job.
            locationUri - Specifies the location of the database
        """

   if (args.size != 8) {
        println(usage)
        exitProcess(1)
   }

   val iam = args[0]
   val s3Path = args[1]
   val cron = args[2]
   val dbName = args[3]
   val crawlerName = args[4]
   val jobName = args[5]
   val scriptLocation = args[6]
```

```
    val locationUri = args[7]

    println("About to start the AWS Glue Scenario")
    createDatabase(dbName, locationUri)
    createCrawler(iam, s3Path, cron, dbName, crawlerName)
    getCrawler(crawlerName)
    startCrawler(crawlerName)
    getDatabase(dbName)
    getGlueTables(dbName)
    createJob(jobName, iam, scriptLocation)
    startJob(jobName)
    getJobs()
    getJobRuns(jobName)
    deleteJob(jobName)
    println("*** Wait for 5 MIN so the $crawlerName is ready to be deleted")
    TimeUnit.MINUTES.sleep(5)
    deleteMyDatabase(dbName)
    deleteCrawler(crawlerName)
}

suspend fun createDatabase(
    dbName: String?,
    locationUriVal: String?,
) {
    val input =
        DatabaseInput {
            description = "Built with the AWS SDK for Kotlin"
            name = dbName
            locationUri = locationUriVal
        }

    val request =
        CreateDatabaseRequest {
            databaseInput = input
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createDatabase(request)
        println("The database was successfully created")
    }
}

suspend fun createCrawler(
    iam: String?,
```

```
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    val targetList = ArrayList<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val crawlerRequest =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Java API"
            targets = targetOb
            role = iam
            schedule = cron
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createCrawler(crawlerRequest)
        println("$crawlerName was successfully created")
    }
}

suspend fun getCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
```

```
    }
}

suspend fun startCrawler(crawlerName: String) {
    val crawlerRequest =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.startCrawler(crawlerRequest)
        println("$crawlerName was successfully started.")
    }
}

suspend fun getDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}

suspend fun getGlueTables(dbName: String?) {
    val tableRequest =
        GetTablesRequest {
            databaseName = dbName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getTables(tableRequest)
        response.tableList?.forEach { tableName ->
            println("Table name is ${tableName.name}")
        }
    }
}

suspend fun startJob(jobNameVal: String?) {
    val runRequest =
```

```
        StartJobRunRequest {
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.startJobRun(runRequest)
        println("The job run Id is ${response.jobRunId}")
    }
}

suspend fun createJob(
    jobName: String,
    iam: String?,
    scriptLocationVal: String?,
) {
    val commandOb =
        JobCommand {
            pythonVersion = "3"
            name = "MyJob1"
            scriptLocation = scriptLocationVal
        }

    val jobRequest =
        CreateJobRequest {
            description = "A Job created by using the AWS SDK for Java V2"
            glueVersion = "2.0"
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            name = jobName
            role = iam
            command = commandOb
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createJob(jobRequest)
        println("$jobName was successfully created.")
    }
}

suspend fun getJobs() {
    val request =
        GetJobsRequest {
```

```
            maxResults = 10
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobs(request)
        response.jobs?.forEach { job ->
            println("Job name is ${job.name}")
        }
    }
}

suspend fun getJobRuns(jobNameVal: String?) {
    val request =
        GetJobRunsRequest {
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobRuns(request)
        response.jobRuns?.forEach { job ->
            println("Job name is ${job.jobName}")
        }
    }
}

suspend fun deleteJob(jobNameVal: String) {
    val jobRequest =
        DeleteJobRequest {
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteJob(jobRequest)
        println("$jobNameVal was successfully deleted")
    }
}

suspend fun deleteMyDatabase(databaseName: String) {
    val request =
        DeleteDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
```

```
        glueClient.deleteDatabase(request)
        println("$databaseName was successfully deleted")
    }
}

suspend fun deleteCrawler(crawlerName: String) {
    val request =
        DeleteCrawlerRequest {
            name = crawlerName
        }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin 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
  - GetJob
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createGlueCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    // Add the S3Target to a list.
    val targetList = mutableListOf<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val request =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Kotlin API"
            targets = targetOb
            role = iam
```

```
            schedule = cron
        }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.createCrawler(request)
        println("$crawlerName was successfully created")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateCrawler」を参照してください。

`GetCrawler`

次の例は、`GetCrawler` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getSpecificCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetCrawler」を参照してください。

GetDatabase

次の例は、GetDatabase を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getSpecificDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetDatabase」を参照してください。

StartCrawler

次の例は、StartCrawler を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun startSpecificCrawler(crawlerName: String?) {
    val request =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.startCrawler(request)
        println("$crawlerName was successfully started.")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「StartCrawler」を参照してください。

SDK for Kotlin を使用した IAM の例

次のコード例は、IAM で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

トピック

- 基本
- アクション

基本

基本を学ぶ

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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。

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <username> <policyName> <roleName> <roleSessionName> <fileLocation>
 <bucketName>

    Where:
        username - The name of the IAM user to create.
        policyName - The name of the policy to create.
        roleName - The name of the role to create.
        roleSessionName - The name of the session required for the assumeRole
 operation.
        fileLocation - The file location to the JSON required to create the role
 (see Readme).
        bucketName - The name of the Amazon S3 bucket from which objects are read.
    """
```

```
    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val userName = args[0]
    val policyName = args[1]
    val roleName = args[2]
    val roleSessionName = args[3]
    val fileLocation = args[4]
    val bucketName = args[5]

    createUser(userName)
    println("$userName was successfully created.")

    val polArn = createPolicy(policyName)
    println("The policy $polArn was successfully created.")

    val roleArn = createRole(roleName, fileLocation)
    println("$roleArn was successfully created.")
    attachRolePolicy(roleName, polArn)

    println("*** Wait for 1 MIN so the resource is available.")
    delay(60000)
    assumeGivenRole(roleArn, roleSessionName, bucketName)

    println("*** Getting ready to delete the AWS resources.")
    deleteRole(roleName, polArn)
    deleteUser(userName)
    println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

```
suspend fun createPolicy(policyNameVal: String?): String {
    val policyDocumentValue: String =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}"

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(
    rolenameVal: String?,
    fileLocation: String?,
): String? {
    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toJSONString()
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
```

```
    }
}

suspend fun attachRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role $roleNameVal")
    }
}

fun checkMyList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
```

```
            return -1
        }
    }
    return 0
}

suspend fun assumeGivenRole(
    roleArnVal: String?,
    roleSessionNameVal: String?,
    bucketName: String,
) {
    val stsClient =
        StsClient {
            region = "us-east-1"
        }

    val roleRequest =
        AssumeRoleRequest {
            roleArn = roleArnVal
            roleSessionName = roleSessionNameVal
        }

    val roleResponse = stsClient.assumeRole(roleRequest)
    val myCreds = roleResponse.credentials
    val key = myCreds?.accessKeyId
    val secKey = myCreds?.secretAccessKey
    val secToken = myCreds?.sessionToken

    val staticCredentials =
        StaticCredentialsProvider {
            accessKeyId = key
            secretAccessKey = secKey
            sessionToken = secToken
        }

    // List all objects in an Amazon S3 bucket using the temp creds.
    val s3 =
        S3Client {
            credentialsProvider = staticCredentials
            region = "us-east-1"
        }

    println("Created a S3Client using temp credentials.")
    println("Listing objects in $bucketName")
```

```
    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }

    val response = s3.listObjects(listObjects)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun deleteRole(
    roleNameVal: String,
    polArn: String,
) {
    val iam = IamClient { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest =
        DetachRolePolicyRequest {
            policyArn = polArn
            roleName = roleNameVal
        }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request =
        DeletePolicyRequest {
            policyArn = polArn
        }

    iam.deletePolicy(request)
    println("*** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest =
        DeleteRoleRequest {
            roleName = roleNameVal
        }

    iam.deleteRole(roleRequest)
```

```
    println("*** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient { region = "AWS_GLOBAL" }
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

- API の詳細については、「AWS SDK for Kotlin 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

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun attachIAMRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
```

```
        println("Successfully attached policy $policyArnVal to role $roleNameVal")
    }
}

fun checkList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「AttachRolePolicy」を参照してください。

`CreateAccessKey`

次の例は、`CreateAccessKey` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createIAMAccessKey(user: String?): String {
    val request =
        CreateAccessKeyRequest {
            userName = user
        }
```

```
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateAccessKey」を参照してください。

`CreateAccountAlias`

次の例は、`CreateAccountAlias` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createIAMAccountAlias(alias: String) {
    val request =
        CreateAccountAliasRequest {
            accountAlias = alias
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateAccountAlias」を参照してください。

`CreatePolicy`

次の例は、`CreatePolicy` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createIAMPolicy(policyNameVal: String?): String {
    val policyDocumentVal =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"dynamodb:DeleteItem\"," +
            "            \"dynamodb:GetItem\"," +
            "            \"dynamodb:PutItem\"," +
            "            \"dynamodb:Scan\"," +
            "            \"dynamodb:UpdateItem\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}"

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreatePolicy」を参照してください。

`CreateUser`

次の例は、`CreateUser` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createIAMUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateUser」を参照してください。

`DeleteAccessKey`

次の例は、`DeleteAccessKey` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteKey(
    userNameVal: String,
    accessKey: String,
) {
    val request =
        DeleteAccessKeyRequest {
            accessKeyId = accessKey
            userName = userNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccessKey(request)
        println("Successfully deleted access key $accessKey from $userNameVal")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteAccessKey」を参照してください。

`DeleteAccountAlias`

次の例は、`DeleteAccountAlias` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteIAMAccountAlias(alias: String) {
    val request =
        DeleteAccountAliasRequest {
            accountAlias = alias
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteAccountAlias」を参照してください。

`DeletePolicy`

次の例は、`DeletePolicy` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {
    val request =
        DeletePolicyRequest {
            policyArn = policyARNVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeletePolicy」を参照してください。

`DeleteUser`

次の例は、`DeleteUser` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteIAMUser(userNameVal: String) {
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    // To delete a user, ensure that the user's access keys are deleted first.
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteUser」を参照してください。

`DetachRolePolicy`

次の例は、`DetachRolePolicy` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun detachPolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        DetachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DetachRolePolicy」を参照してください。

`GetPolicy`

次の例は、`GetPolicy` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getIAMPolicy(policyArnVal: String?) {
    val request =
        GetPolicyRequest {
            policyArn = policyArnVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetPolicy」を参照してください。

`ListAccessKeys`

次の例は、`ListAccessKeys` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listKeys(userNameVal: String?) {
    val request =
        ListAccessKeysRequest {
            userName = userNameVal
        }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ListAccessKeys」を参照してください。

`ListAccountAliases`

次の例は、`ListAccountAliases` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listAliases() {
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ListAccountAliases」を参照してください。

`ListUsers`

次の例は、`ListUsers` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listAllUsers() {
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listUsers(ListUsersRequest { })
        response.users?.forEach { user ->
            println("Retrieved user ${user.userName}")
            val permissionsBoundary = user.permissionsBoundary
            if (permissionsBoundary != null) {
                println("Permissions boundary details
 ${permissionsBoundary.permissionsBoundaryType}")
            }
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ListUsers」を参照してください。

`UpdateUser`

次の例は、`UpdateUser` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun updateIAMUser(
```

```
    curName: String?,
    newName: String?,
) {
    val request =
        UpdateUserRequest {
            userName = curName
            newUserName = newName
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.updateUser(request)
        println("Successfully updated user to $newName")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「UpdateUser」を参照してください。

AWS IoT SDK for Kotlin を使用した の例

次のコード例は、 で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

こんにち 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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.ListThingsRequest

suspend fun main() {
    println("A listing of your AWS IoT Things:")
    listAllThings()
}

suspend fun listAllThings() {
    val thingsRequest =
        ListThingsRequest {
            maxResults = 10
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listThings(thingsRequest)
        val thingList = response.things
        if (thingList != null) {
            for (attribute in thingList) {
                println("Thing name ${attribute.thingName}")
                println("Thing ARN: ${attribute.thingArn}")
            }
        }
    }
}
```

- API の詳細については、[AWS SDK for Kotlin API リファレンス]の「listThings」を参照してください。

トピック

- 基本

- アクション

基本

基本を学ぶ

次のコードサンプルは、以下の操作方法を示しています。

- AWS IoT モノを作成します。
- デバイス証明書を生成します。
- 属性を使用して AWS IoT モノを更新します。
- 一意のエンドポイントを返します。
- AWS IoT 証明書を一覧表示します。
- AWS IoT シャドウを作成します。
- 状態情報を書き込みます。
- ルールを作成します。
- ルールを一覧表示します。
- モノの名前を使用してモノを検索します。
- AWS IoT モノを削除します。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.Action
import aws.sdk.kotlin.services.iot.model.AttachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.AttributePayload
import aws.sdk.kotlin.services.iot.model.CreateThingRequest
import aws.sdk.kotlin.services.iot.model.CreateTopicRuleRequest
import aws.sdk.kotlin.services.iot.model.DeleteCertificateRequest
import aws.sdk.kotlin.services.iot.model.DeleteThingRequest
```

```
import aws.sdk.kotlin.services.iot.model.DescribeEndpointRequest
import aws.sdk.kotlin.services.iot.model.DescribeThingRequest
import aws.sdk.kotlin.services.iot.model.DetachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.ListTopicRulesRequest
import aws.sdk.kotlin.services.iot.model.SearchIndexRequest
import aws.sdk.kotlin.services.iot.model.SnsAction
import aws.sdk.kotlin.services.iot.model.TopicRulePayload
import aws.sdk.kotlin.services.iot.model.UpdateThingRequest
import aws.sdk.kotlin.services.iotdataplane.IotDataPlaneClient
import aws.sdk.kotlin.services.iotdataplane.model.GetThingShadowRequest
import aws.sdk.kotlin.services.iotdataplane.model.UpdateThingShadowRequest
import aws.smithy.kotlin.runtime.content.ByteStream
import aws.smithy.kotlin.runtime.content.toByteArray
import java.util.Scanner
import java.util.regex.Pattern
import kotlin.system.exitProcess

/**
 * Before running this Kotlin code example, ensure that your development environment
 * is set up, including configuring your credentials.
 *
 * For detailed instructions, refer to the following documentation topic:
 * [Setting Up Your Development Environment](https://docs.aws.amazon.com/sdk-for-
kotlin/latest/developer-guide/setup.html)
 *
 * This code example requires an SNS topic and an IAM Role.
 * Follow the steps in the documentation to set up these resources:
 *
 * - [Creating an SNS Topic](https://docs.aws.amazon.com/sns/latest/dg/sns-getting-
started.html#step-create-topic)
 * - [Creating an IAM Role](https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html)
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")
val TOPIC = "your-iot-topic"

suspend fun main(args: Array<String>) {
    val 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.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    var thingName: String
    val roleARN = args[0]
    val snsAction = args[1]
    val scanner = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the AWS IoT example scenario.")
    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 Kotlin 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 Kotlin environment.
        """.trimIndent(),
    )

    print("Press Enter to continue...")
    scanner.nextLine()
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS IoT thing.")
    println(
        """
        An AWS IoT thing represents a virtual entity in the AWS IoT service that can
be associated with a physical device.
```

```
        """.trimIndent(),
    )
    // Prompt the user for input.
    print("Enter thing name: ")
    thingName = scanner.nextLine()
    createIoTThing(thingName)
    describeThing(thingName)
    println(DASHES)

    println(DASHES)
    println("2. Generate a device certificate.")
    println(
        """
        A device certificate performs a role in securing the communication between
devices (things) and the AWS IoT platform.
        """.trimIndent(),
    )

    print("Do you want to create a certificate for $thingName? (y/n)")
    val certAns = scanner.nextLine()
    var certificateArn: String? = ""
    if (certAns != null && certAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
        certificateArn = createCertificate()
        println("Attach the certificate to the AWS IoT thing.")
        attachCertificateToThing(thingName, certificateArn)
    } else {
        println("A device certificate was not created.")
    }
    println(DASHES)

    println(DASHES)
    println("3. Update an AWS IoT thing with Attributes.")
    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.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateThing(thingName)
    println(DASHES)
```

```
    println(DASHES)
    println("4. Return a unique endpoint specific to the Amazon Web Services
account.")
    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.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    val endpointUrl = describeEndpoint()
    println(DASHES)

    println(DASHES)
    println("5. List your AWS IoT certificates")
    print("Press Enter to continue...")
    scanner.nextLine()
    if (certificateArn!!.isNotEmpty()) {
        listCertificates()
    } else {
        println("You did not create a certificates. Skipping this step.")
    }
    println(DASHES)

    println(DASHES)
    println("6. Create an IoT shadow that refers to a digital representation or
virtual twin of a physical IoT device")
    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.

        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateShawdowThing(thingName)
```

```
    println(DASHES)

    println(DASHES)
    println("7. Write out the state information, in JSON format.")
    print("Press Enter to continue...")
    scanner.nextLine()
    getPayload(thingName)
    println(DASHES)

    println(DASHES)
    println("8. Creates a rule")
    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.
        """.trimIndent(),
    )
    print("Enter Rule name: ")
    val ruleName = scanner.nextLine()
    createIoTRule(roleARN, ruleName, snsAction)
    println(DASHES)

    println(DASHES)
    println("9. List your rules.")
    print("Press Enter to continue...")
    scanner.nextLine()
    listIoTRules()
    println(DASHES)

    println(DASHES)
    println("10. Search things using the name.")
    print("Press Enter to continue...")
    scanner.nextLine()
    val queryString = "thingName:$thingName"
    searchThings(queryString)
    println(DASHES)

    println(DASHES)
    if (certificateArn.length > 0) {
        print("Do you want to detach and delete the certificate for $thingName? (y/
n)")
        val delAns = scanner.nextLine()
```

```
        if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
 true)) {
            println("11. You selected to detach amd delete the certificate.")
            print("Press Enter to continue...")
            scanner.nextLine()
            detachThingPrincipal(thingName, certificateArn)
            deleteCertificate(certificateArn)
        } else {
            println("11. You selected not to delete the certificate.")
        }
    } else {
        println("11. You did not create a certificate so there is nothing to
 delete.")
    }
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS IoT thing.")
    print("Do you want to delete the IoT thing? (y/n)")
    val delAns = scanner.nextLine()
    if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase = true))
 {
        deleteIoTThing(thingName)
    } else {
        println("The IoT thing was not deleted.")
    }
    println(DASHES)

    println(DASHES)
    println("The AWS IoT workflow has successfully completed.")
    println(DASHES)
}

suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}

private fun extractCertificateId(certificateArn: String): String? {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
    val arnParts = certificateArn.split(":".toRegex()).dropLastWhile
 { it.isEmpty() }.toTypedArray()
    val certificateIdPart = arnParts[arnParts.size - 1]
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1)
}

suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}

suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
```

```
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}

suspend fun listIoTRules() {
    val listTopicRulesRequest = ListTopicRulesRequest {}

    IotClient { region = "us-east-1" }.use { iotClient ->
        val listTopicRulesResponse = iotClient.listTopicRules(listTopicRulesRequest)
        println("List of IoT rules:")
        val ruleList = listTopicRulesResponse.rules
        ruleList?.forEach { rule ->
            println("Rule name: ${rule.ruleName}")
            println("Rule ARN: ${rule.ruleArn}")
            println("--------------")
        }
    }
}

suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
```

```
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}

suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}

suspend fun listCertificates() {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}

suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
```

```
    IotClient { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}

private fun getValue(input: String?): String {
    // Define a regular expression pattern for extracting the subdomain.
    val pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\\.com")

    // Match the pattern against the input string.
    val matcher = pattern.matcher(input)

    // Check if a match is found.
    if (matcher.find()) {
        val subdomain = matcher.group(1)
        println("Extracted subdomain: $subdomain")
        return subdomain
    } else {
        println("No match found")
    }
    return ""
}

suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
```

```
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}

suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }

    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}

suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

```
suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}

suspend fun createCertificate(): String? {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}

suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「AttachThingPrincipal」を参照してください。

`CreateKeysAndCertificate`

次の例は、`CreateKeysAndCertificate` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createCertificate(): String? {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateKeysAndCertificate」を参照してください。

`CreateThing`

次の例は、`CreateThing` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createIoTThing(thingNameVal: String) {
```

```
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateThing」を参照してください。

`CreateTopicRule`

次の例は、`CreateTopicRule` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
```

```
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateTopicRule」を参照してください。

`DeleteCertificate`

次の例は、`DeleteCertificate` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
```

```
        }
    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteCertificate」を参照してください。

`DeleteThing`

次の例は、DeleteThing を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteThing」を参照してください。

`DescribeEndpoint`

次の例は、`DescribeEndpoint` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeEndpoint」を参照してください。

`DescribeThing`

次の例は、`DescribeThing` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeThing」を参照してください。

`DetachThingPrincipal`

次の例は、`DetachThingPrincipal` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }
```

```
    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DetachThingPrincipal」を参照してください。

`ListCertificates`

次の例は、`ListCertificates` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listCertificates() {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ListCertificates」を参照してください。

`SearchIndex`

次の例は、`SearchIndex` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「SearchIndex」を参照してください。

`UpdateThing`

次の例は、`UpdateThing` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「UpdateThing」を参照してください。

AWS IoT data SDK for Kotlin を使用した の例

次のコード例は、 で AWS SDK for Kotlin 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を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのGetThingShadow」を参照してください。

UpdateThingShadow

次の例は、UpdateThingShadow を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }

    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのUpdateThingShadow」を参照してください。

AWS IoT FleetWise SDK for Kotlin を使用した の例

次のコード例は、 で AWS SDK for Kotlin 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 FleetWise。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

こんにち AWS IoT FleetWiseは

次のコード例は、 AWS IoT FleetWiseの使用を開始する方法を示しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listSignalCatalogs()
}

/**
 * Lists the AWS FleetWise Signal Catalogs associated with the current AWS account.
 */
suspend fun listSignalCatalogs() {
    val request = ListSignalCatalogsRequest {
        maxResults = 10
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        val response = fleetwiseClient.listSignalCatalogs(request)
```

```
        val summaries = response.summaries

        if (summaries.isNullOrEmpty()) {
            println("No AWS FleetWise Signal Catalogs were found.")
        } else {
            summaries.forEach { summary ->
                with(summary) {
                    println("Catalog Name: $name")
                    println("ARN: $arn")
                    println("Created: $creationTime")
                    println("Last Modified: $lastModificationTime")
                    println("---------------")
                }
            }
        }
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのlistSignalCatalogsPaginator」を参照してください。

トピック

- 基本
- アクション

基本

基本を学ぶ

次のコードサンプルは、以下の操作方法を示しています。

- 標準化されたシグナルのコレクションを作成します。
- 車両グループを表すフリートを作成します。
- モデルマニフェストを作成します。
- デコーダーマニフェストを作成します。
- モデルマニフェストのステータスを確認します。
- デコーダーのステータスを確認します。

- IoT モノを作成します。
- 車両を作成します。
- 車両の詳細を表示します。
- AWS IoT FleetWise アセットを削除します。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

AWS IoT SiteWise 機能を示すインタラクティブなシナリオを実行します。

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
var scanner = Scanner(System.`in`)
val DASHES = String(CharArray(80)).replace("\u0000", "-")
suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage:
            <signalCatalogName> <manifestName> <fleetId> <vecName> <decName>

        Where:
            signalCatalogName     - The name of the Signal Catalog to create (eg,
 catalog30).
            manifestName          - The name of the Vehicle Model (Model Manifest)
 to create (eg, manifest30).
            fleetId               - The ID of the Fleet to create (eg, fleet30).
            vecName               - The name of the Vehicle to create (eg,
 vehicle30).
            decName               - The name of the Decoder Manifest to create (eg,
 decManifest30).
```

```
        """.trimIndent()

    if (args.size != 5) {
        println(usage)
        return
    }

    val signalCatalogName = args[0]
    val manifestName = args[1]
    val fleetId = args[2]
    val vecName = args[3]
    val decName = args[4]

    println(
        """
        AWS IoT FleetWise is a managed service that simplifies the
        process of collecting, organizing, and transmitting vehicle
        data to the cloud in near real-time. Designed for automakers
        and fleet operators, it allows you to define vehicle models,
        specify the exact data you want to collect (such as engine
        temperature, speed, or battery status), and send this data to
        AWS for analysis. By using intelligent data collection
        techniques, IoT FleetWise reduces the volume of data
        transmitted by filtering and transforming it at the edge,
        helping to minimize bandwidth usage and costs.

        At its core, AWS IoT FleetWise helps organizations build
        scalable systems for vehicle data management and analytics,
        supporting a wide variety of vehicles and sensor configurations.
        You can define signal catalogs and decoder manifests that describe
        how raw CAN bus signals are translated into readable data, making
        the platform highly flexible and extensible. This allows
        manufacturers to optimize vehicle performance, improve safety,
        and reduce maintenance costs by gaining real-time visibility
        into fleet operations.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    println(DASHES)
    runScenario(signalCatalogName, fleetId, manifestName, decName, vecName)
}

suspend fun runScenario(signalCatalogName: String, fleetIdVal: String, manifestName:
 String, decName: String, vecName: String) {
```

```
    println(DASHES)
    println("1. Creates a collection of standardized signals that can be reused to
create vehicle models")
    waitForInputToContinue(scanner)
    val signalCatalogArn = createbranchVehicle(signalCatalogName)
    println("The collection ARN is $signalCatalogArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("2. Create a fleet that represents a group of vehicles")
    println(
        """
        Creating an IoT FleetWise fleet allows you to efficiently collect,
        organize, and transfer vehicle data to the cloud, enabling real-time
        insights into vehicle performance and health.

        It helps reduce data costs by allowing you to filter and prioritize
        only the most relevant vehicle signals, supporting advanced analytics
        and predictive maintenance use cases.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val fleetid = createFleet(signalCatalogArn, fleetIdVal)
    println("The fleet Id is $fleetid")
    waitForInputToContinue(scanner)
    val nodeList = listSignalCatalogNode(signalCatalogName)
    println(DASHES)

    println(DASHES)
    println("3. Create a model manifest")
    println(
        """
        An AWS IoT FleetWise manifest defines the structure and
        relationships of vehicle data. The model manifest specifies
        which signals to collect and how they relate to vehicle systems,
        while the decoder manifest defines how to decode raw vehicle data
        into meaningful signals.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val nodes = listSignalCatalogNode(signalCatalogName)
    val manifestArn = nodes?.let { createModelManifest(manifestName,
signalCatalogArn, it) }
```

```
    println("The manifest ARN is $manifestArn")
    println(DASHES)

    println(DASHES)
    println("4. Create a decoder manifest")
    println(
        """
        A decoder manifest in AWS IoT FleetWise defines how raw vehicle
        data (such as CAN signals) should be interpreted and decoded
        into meaningful signals. It acts as a translation layer
        that maps vehicle-specific protocols to standardized data formats
        using decoding rules. This is crucial for extracting usable
        data from different vehicle models, even when their data
        formats vary.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val decArn = createDecoderManifest(decName, manifestArn)
    println("The decoder manifest ARN is $decArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("5. Check the status of the model manifest")
    println(
        """
        The model manifest must be in an ACTIVE state before it can be used
        to create or update a vehicle.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    updateModelManifest(manifestName)
    waitForModelManifestActive(manifestName)
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("6. Check the status of the decoder")
    println(
        """
        The decoder manifest must be in an ACTIVE state before it can be used
        to create or update a vehicle.
        """.trimIndent(),
    )
```

```
    waitForInputToContinue(scanner)
    updateDecoderManifest(decName)
    waitForDecoderManifestActive(decName)
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("7. Create an IoT Thing")
    println(
        """
        AWS IoT FleetWise expects an existing AWS IoT Thing with the same
        name as the vehicle name you are passing to createVehicle method.
        Before calling createVehicle(), you must create an AWS IoT Thing
        with the same name using the AWS IoT Core service.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    createThingIfNotExist(vecName)
    println(DASHES)

    println(DASHES)
    println("8. Create a vehicle")
    println(
        """
        Creating a vehicle in AWS IoT FleetWise allows you to digitally
        represent and manage a physical vehicle within the AWS ecosystem.
        This enables efficient ingestion, transformation, and transmission
        of vehicle telemetry data to the cloud for analysis.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    createVehicle(vecName, manifestArn, decArn)
    println(DASHES)

    println(DASHES)
    println("9. Display vehicle details")
    waitForInputToContinue(scanner)
    getVehicleDetails(vecName)
    waitForInputToContinue(scanner)
    println(DASHES)
    println(DASHES)
    println("10. Delete the AWS IoT Fleetwise Assets")
    println("Would you like to delete the IoT Fleetwise Assets? (y/n)")
    val delAns = scanner.nextLine().trim()
```

```
    if (delAns.equals("y", ignoreCase = true)) {
        deleteVehicle(vecName)
        deleteDecoderManifest(decName)
        deleteModelManifest(manifestName)
        deleteFleet(fleetid)
        deleteSignalCatalog(signalCatalogName)
    }

    println(DASHES)
    println(
        """
        Thank you for checking out the AWS IoT Fleetwise Service Use demo. We hope
 you
        learned something new, or got some inspiration for your own apps today.
        For more AWS code examples, have a look at:
        https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-library.html
        """.trimIndent(),
    )
    println(DASHES)
}

suspend fun deleteVehicle(vecName: String) {
    val request = DeleteVehicleRequest {
        vehicleName = vecName
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        fleetwiseClient.deleteVehicle(request)
        println("Vehicle $vecName was deleted successfully.")
    }
}

suspend fun getVehicleDetails(vehicleNameVal: String) {
    val request = GetVehicleRequest {
        vehicleName = vehicleNameVal
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        val response = fleetwiseClient.getVehicle(request)
        val details = mapOf(
            "vehicleName" to response.vehicleName,
            "arn" to response.arn,
            "modelManifestArn" to response.modelManifestArn,
            "decoderManifestArn" to response.decoderManifestArn,
```

```
            "attributes" to response.attributes.toString(),
            "creationTime" to response.creationTime.toString(),
            "lastModificationTime" to response.lastModificationTime.toString(),
        )

        println("Vehicle Details:")
        for ((key, value) in details) {
            println("• %-20s : %s".format(key, value))
        }
    }
}

suspend fun createVehicle(vecName: String, manifestArn: String?, decArn: String) {
    val request = CreateVehicleRequest {
        vehicleName = vecName
        modelManifestArn = manifestArn
        decoderManifestArn = decArn
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        fleetwiseClient.createVehicle(request)
        println("Vehicle $vecName was created successfully.")
    }
}

/**
 * Creates an IoT Thing if it does not already exist.
 *
 * @param vecName the name of the IoT Thing to create
 */
suspend fun createThingIfNotExist(vecName: String) {
    val request = CreateThingRequest {
        thingName = vecName
    }

    IotClient { region = "us-east-1" }.use { client ->
        client.createThing(request)
        println("The $vecName IoT Thing was successfully created")
    }
}

suspend fun updateDecoderManifest(nameVal: String) {
    val request = UpdateDecoderManifestRequest {
        name = nameVal
```

```
        status = ManifestStatus.Active
    }
    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        fleetwiseClient.updateDecoderManifest(request)
        println("$nameVal was successfully updated")
    }
}

/**
 * Waits for the specified model manifest to become active.
 *
 * @param decNameVal the name of the model manifest to wait for
 */
suspend fun waitForDecoderManifestActive(decNameVal: String) {
    var elapsedSeconds = 0
    var lastStatus: ManifestStatus = ManifestStatus.Draft

    print("# Elapsed: 0s | Status: DRAFT")
    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        while (true) {
            delay(1000)
            elapsedSeconds++
            if (elapsedSeconds % 5 == 0) {
                val request = GetDecoderManifestRequest {
                    name = decNameVal
                }

                val response = fleetwiseClient.getDecoderManifest(request)
                lastStatus = response.status ?: ManifestStatus.Draft

                when (lastStatus) {
                    ManifestStatus.Active -> {
                        print("\rElapsed: ${elapsedSeconds}s | Status: ACTIVE #\n")
                        return
                    }

                    ManifestStatus.Invalid -> {
                        print("\rElapsed: ${elapsedSeconds}s | Status: INVALID #\n")
                        throw RuntimeException("Model manifest became INVALID.
 Cannot proceed.")
                    }

                    else -> {
```

```
                        print("\r Elapsed: ${elapsedSeconds}s | Status:
 $lastStatus")
                    }
                }
            } else {
                print("\r Elapsed: ${elapsedSeconds}s | Status: $lastStatus")
            }
        }
    }
}

/**
 * Waits for the specified model manifest to become active.
 *
 * @param manifestName the name of the model manifest to wait for
 */
suspend fun waitForModelManifestActive(manifestNameVal: String) {
    var elapsedSeconds = 0
    var lastStatus: ManifestStatus = ManifestStatus.Draft

    print("# Elapsed: 0s | Status: DRAFT")
    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        while (true) {
            delay(1000)
            elapsedSeconds++
            if (elapsedSeconds % 5 == 0) {
                val request = GetModelManifestRequest {
                    name = manifestNameVal
                }

                val response = fleetwiseClient.getModelManifest(request)
                lastStatus = response.status ?: ManifestStatus.Draft

                when (lastStatus) {
                    ManifestStatus.Active -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status: ACTIVE #\n")
                        return
                    }

                    ManifestStatus.Invalid -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status: INVALID #
\n")
                        throw RuntimeException("Model manifest became INVALID.
 Cannot proceed.")
```

```
                    }

                    else -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status:
 $lastStatus")
                    }
                }
            } else {
                print("\r Elapsed: ${elapsedSeconds}s | Status: $lastStatus")
            }
        }
    }
}

/**
 * Updates the model manifest.
 *
 * @param nameVal the name of the model manifest to update
 */
suspend fun updateModelManifest(nameVal: String) {
    val request = UpdateModelManifestRequest {
        name = nameVal
        status = ManifestStatus.Active
    }
    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        fleetwiseClient.updateModelManifest(request)
        println("$nameVal was successfully updated")
    }
}

suspend fun deleteDecoderManifest(nameVal: String) {
    val request = DeleteDecoderManifestRequest {
        name = nameVal
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        fleetwiseClient.deleteDecoderManifest(request)
        println("$nameVal was successfully deleted")
    }
}

/**
 * Creates a new decoder manifest.
 *
```

```
 * @param decName             the name of the decoder manifest
 * @param modelManifestArnVal the ARN of the model manifest
 * @return the ARN of the decoder manifest
 */
suspend fun createDecoderManifest(decName: String, modelManifestArnVal: String?):
 String {
    val interfaceIdVal = "can0"

    val canInter = CanInterface {
        name = "canInterface0"
        protocolName = "CAN"
        protocolVersion = "1.0"
    }

    val networkInterface = NetworkInterface {
        interfaceId = interfaceIdVal
        type = NetworkInterfaceType.CanInterface
        canInterface = canInter
    }

    val carRpmSig = CanSignal {
        messageId = 100
        isBigEndian = false
        isSigned = false
        startBit = 16
        length = 16
        factor = 1.0
        offset = 0.0
    }

    val carSpeedSig = CanSignal {
        messageId = 101
        isBigEndian = false
        isSigned = false
        startBit = 0
        length = 16
        factor = 1.0
        offset = 0.0
    }

    val engineRpmDecoder = SignalDecoder {
        fullyQualifiedName = "Vehicle.Powertrain.EngineRPM"
        interfaceId = interfaceIdVal
        type = SignalDecoderType.CanSignal
```

```
        canSignal = carRpmSig
    }

    val vehicleSpeedDecoder = SignalDecoder {
        fullyQualifiedName = "Vehicle.Powertrain.VehicleSpeed"
        interfaceId = interfaceIdVal
        type = SignalDecoderType.CanSignal
        canSignal = carSpeedSig
    }

    val request = CreateDecoderManifestRequest {
        name = decName
        modelManifestArn = modelManifestArnVal
        networkInterfaces = listOf(networkInterface)
        signalDecoders = listOf(engineRpmDecoder, vehicleSpeedDecoder)
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        val response = fleetwiseClient.createDecoderManifest(request)
        return response.arn
    }
}

/**
 * Deletes a signal catalog.
 *
 * @param name the name of the signal catalog to delete
 */
suspend fun deleteSignalCatalog(catName: String) {
    val request = DeleteSignalCatalogRequest {
        name = catName
    }
    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        fleetwiseClient.deleteSignalCatalog(request)
        println(" $catName was successfully deleted")
    }
}

/**
 * Deletes a fleet based on the provided fleet ID.
 *
 * @param fleetId the ID of the fleet to be deleted
 */
suspend fun deleteFleet(fleetIdVal: String) {
```

```
    val request = DeleteFleetRequest {
        fleetId = fleetIdVal
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        fleetwiseClient.deleteFleet(request)
        println(" $fleetIdVal was successfully deleted")
    }
}

/**
 * Deletes a model manifest.
 *
 * @param nameVal the name of the model manifest to delete
 */
suspend fun deleteModelManifest(nameVal: String) {
    val request = DeleteModelManifestRequest {
        name = nameVal
    }
    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        fleetwiseClient.deleteModelManifest(request)
        println(" $nameVal was successfully deleted")
    }
}

/**
 * Creates a model manifest.
 *
 * @param name              the name of the model manifest to create
 * @param signalCatalogArn  the Amazon Resource Name (ARN) of the signal catalog
 * @param nodes             a list of nodes to include in the model manifest
 * @return a {@link CompletableFuture} that completes with the ARN of the created
 model manifest
 */
suspend fun createModelManifest(nameVal: String, signalCatalogArnVal: String,
 nodesList: List<Node>): String {
    val fqnList: List<String> = nodesList.map { node ->
        when (node) {
            is Node.Sensor -> node.asSensor().fullyQualifiedName
            is Node.Branch -> node.asBranch().fullyQualifiedName
            else -> throw RuntimeException("Unsupported node type")
        }
    }
```

```
    val request = CreateModelManifestRequest {
        name = nameVal
        signalCatalogArn = signalCatalogArnVal
        nodes = fqnList
    }
    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        val response = fleetwiseClient.createModelManifest(request)
        return response.arn
    }
}

/**
 * Lists the signal catalog nodes asynchronously.
 *
 * @param signalCatalogName the name of the signal catalog
 * @return a CompletableFuture that, when completed, contains a list of nodes in the
 specified signal catalog
 * @throws CompletionException if an exception occurs during the asynchronous
 operation
 */
suspend fun listSignalCatalogNode(signalCatalogName: String): List<Node>? {
    val request = ListSignalCatalogNodesRequest {
        name = signalCatalogName
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        val response = fleetwiseClient.listSignalCatalogNodes(request)
        return response.nodes
    }
}

/**
 * Creates a new fleet.
 *
 * @param catARN the Amazon Resource Name (ARN) of the signal catalog to associate
 with the fleet
 * @param fleetId the unique identifier for the fleet
 * @return the ID of the created fleet
 */
suspend fun createFleet(catARN: String, fleetIdVal: String): String {
    val fleetRequest = CreateFleetRequest {
        fleetId = fleetIdVal
        signalCatalogArn = catARN
        description = "Built using the AWS For Kotlin"
```

```
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        val response = fleetwiseClient.createFleet(fleetRequest)
        return response.id
    }
}

/**
 * Creates a signal catalog.
 *
 * @param signalCatalogName the name of the signal catalog to create the branch
 vehicle in
 * @return the ARN (Amazon Resource Name) of the created signal catalog
 */
suspend fun createbranchVehicle(signalCatalogName: String): String {
    delay(2000) // Wait for 2 seconds
    val branchVehicle = Branch {
        fullyQualifiedName = "Vehicle"
        description = "Root branch"
    }

    val branchPowertrain = Branch {
        fullyQualifiedName = "Vehicle.Powertrain"
        description = "Powertrain branch"
    }

    val sensorRPM = Sensor {
        fullyQualifiedName = "Vehicle.Powertrain.EngineRPM"
        description = "Engine RPM"
        dataType = NodeDataType.Double
        unit = "rpm"
    }

    val sensorKM = Sensor {
        fullyQualifiedName = "Vehicle.Powertrain.VehicleSpeed"
        description = "Vehicle Speed"
        dataType = NodeDataType.Double
        unit = "km/h"
    }

    // Wrap each specific node type (Branch and Sensor) into the sealed Node class
    // so they can be included in the CreateSignalCatalogRequest.
    val myNodes = listOf(
```

```
        Node.Branch(branchVehicle),
        Node.Branch(branchPowertrain),
        Node.Sensor(sensorRPM),
        Node.Sensor(sensorKM),
    )

    val request = CreateSignalCatalogRequest {
        name = signalCatalogName
        nodes = myNodes
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        val response = fleetwiseClient.createSignalCatalog(request)
        return response.arn
    }
}

private fun waitForInputToContinue(scanner: Scanner) {
    while (true) {
        println("")
        println("Enter 'c' followed by <ENTER> to continue:")
        val input = scanner.nextLine()

        if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else {
            println("Invalid input. Please try again.")
        }
    }
}
```

アクション

createDecoderManifest

次の例は、createDecoderManifest を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Creates a new decoder manifest.
 *
 * @param decName             the name of the decoder manifest
 * @param modelManifestArnVal the ARN of the model manifest
 * @return the ARN of the decoder manifest
 */
suspend fun createDecoderManifest(decName: String, modelManifestArnVal: String?):
 String {
    val interfaceIdVal = "can0"

    val canInter = CanInterface {
        name = "canInterface0"
        protocolName = "CAN"
        protocolVersion = "1.0"
    }

    val networkInterface = NetworkInterface {
        interfaceId = interfaceIdVal
        type = NetworkInterfaceType.CanInterface
        canInterface = canInter
    }

    val carRpmSig = CanSignal {
        messageId = 100
        isBigEndian = false
        isSigned = false
        startBit = 16
        length = 16
        factor = 1.0
        offset = 0.0
    }

    val carSpeedSig = CanSignal {
```

```
        messageId = 101
        isBigEndian = false
        isSigned = false
        startBit = 0
        length = 16
        factor = 1.0
        offset = 0.0
    }

    val engineRpmDecoder = SignalDecoder {
        fullyQualifiedName = "Vehicle.Powertrain.EngineRPM"
        interfaceId = interfaceIdVal
        type = SignalDecoderType.CanSignal
        canSignal = carRpmSig
    }

    val vehicleSpeedDecoder = SignalDecoder {
        fullyQualifiedName = "Vehicle.Powertrain.VehicleSpeed"
        interfaceId = interfaceIdVal
        type = SignalDecoderType.CanSignal
        canSignal = carSpeedSig
    }

    val request = CreateDecoderManifestRequest {
        name = decName
        modelManifestArn = modelManifestArnVal
        networkInterfaces = listOf(networkInterface)
        signalDecoders = listOf(engineRpmDecoder, vehicleSpeedDecoder)
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        val response = fleetwiseClient.createDecoderManifest(request)
        return response.arn
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのcreateDecoderManifest」を参照してください。

`createFleet`

次の例は、createFleet を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Creates a new fleet.
 *
 * @param catARN the Amazon Resource Name (ARN) of the signal catalog to associate
 with the fleet
 * @param fleetId the unique identifier for the fleet
 * @return the ID of the created fleet
 */
suspend fun createFleet(catARN: String, fleetIdVal: String): String {
    val fleetRequest = CreateFleetRequest {
        fleetId = fleetIdVal
        signalCatalogArn = catARN
        description = "Built using the AWS For Kotlin"
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        val response = fleetwiseClient.createFleet(fleetRequest)
        return response.id
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのcreateFleet」を参照してください。

`createModelManifest`

次の例は、`createModelManifest` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Creates a model manifest.
 *
 * @param name              the name of the model manifest to create
 * @param signalCatalogArn  the Amazon Resource Name (ARN) of the signal catalog
 * @param nodes             a list of nodes to include in the model manifest
 * @return a {@link CompletableFuture} that completes with the ARN of the created
 model manifest
 */
suspend fun createModelManifest(nameVal: String, signalCatalogArnVal: String,
 nodesList: List<Node>): String {
    val fqnList: List<String> = nodesList.map { node ->
        when (node) {
            is Node.Sensor -> node.asSensor().fullyQualifiedName
            is Node.Branch -> node.asBranch().fullyQualifiedName
            else -> throw RuntimeException("Unsupported node type")
        }
    }

    val request = CreateModelManifestRequest {
        name = nameVal
        signalCatalogArn = signalCatalogArnVal
        nodes = fqnList
    }
    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        val response = fleetwiseClient.createModelManifest(request)
        return response.arn
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのcreateModelManifest」を参照してください。

createSignalCatalog

次の例は、createSignalCatalog を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Creates a signal catalog.
 *
 * @param signalCatalogName the name of the signal catalog to create the branch
 vehicle in
 * @return the ARN (Amazon Resource Name) of the created signal catalog
 */
suspend fun createbranchVehicle(signalCatalogName: String): String {
    delay(2000) // Wait for 2 seconds
    val branchVehicle = Branch {
        fullyQualifiedName = "Vehicle"
        description = "Root branch"
    }

    val branchPowertrain = Branch {
        fullyQualifiedName = "Vehicle.Powertrain"
        description = "Powertrain branch"
    }

    val sensorRPM = Sensor {
        fullyQualifiedName = "Vehicle.Powertrain.EngineRPM"
        description = "Engine RPM"
        dataType = NodeDataType.Double
        unit = "rpm"
    }

    val sensorKM = Sensor {
        fullyQualifiedName = "Vehicle.Powertrain.VehicleSpeed"
        description = "Vehicle Speed"
        dataType = NodeDataType.Double
        unit = "km/h"
```

```
    }

    // Wrap each specific node type (Branch and Sensor) into the sealed Node class
    // so they can be included in the CreateSignalCatalogRequest.
    val myNodes = listOf(
        Node.Branch(branchVehicle),
        Node.Branch(branchPowertrain),
        Node.Sensor(sensorRPM),
        Node.Sensor(sensorKM),
    )

    val request = CreateSignalCatalogRequest {
        name = signalCatalogName
        nodes = myNodes
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        val response = fleetwiseClient.createSignalCatalog(request)
        return response.arn
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのcreateSignalCatalog」を参照してください。

`createVehicle`

次の例は、`createVehicle` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createVehicle(vecName: String, manifestArn: String?, decArn: String) {
    val request = CreateVehicleRequest {
        vehicleName = vecName
```

```
        modelManifestArn = manifestArn
        decoderManifestArn = decArn
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        fleetwiseClient.createVehicle(request)
        println("Vehicle $vecName was created successfully.")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスの「createVehicle」を参照してください。

`deleteDecoderManifest`

次の例は、`deleteDecoderManifest` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteDecoderManifest(nameVal: String) {
    val request = DeleteDecoderManifestRequest {
        name = nameVal
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        fleetwiseClient.deleteDecoderManifest(request)
        println("$nameVal was successfully deleted")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスの「deleteDecoderManifest」を参照してください。

deleteFleet

次の例は、deleteFleet を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Deletes a fleet based on the provided fleet ID.
 *
 * @param fleetId the ID of the fleet to be deleted
 */
suspend fun deleteFleet(fleetIdVal: String) {
    val request = DeleteFleetRequest {
        fleetId = fleetIdVal
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        fleetwiseClient.deleteFleet(request)
        println(" $fleetIdVal was successfully deleted")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのdeleteFleet」を参照してください。

deleteModelManifest

次の例は、deleteModelManifest を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Deletes a model manifest.
 *
 * @param nameVal the name of the model manifest to delete
 */
suspend fun deleteModelManifest(nameVal: String) {
    val request = DeleteModelManifestRequest {
        name = nameVal
    }
    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        fleetwiseClient.deleteModelManifest(request)
        println(" $nameVal was successfully deleted")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのdeleteModelManifest」を参照してください。

`deleteSignalCatalog`

次の例は、`deleteSignalCatalog` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
```

```
 * Deletes a signal catalog.
 *
 * @param name the name of the signal catalog to delete
 */
suspend fun deleteSignalCatalog(catName: String) {
    val request = DeleteSignalCatalogRequest {
        name = catName
    }
    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        fleetwiseClient.deleteSignalCatalog(request)
        println(" $catName was successfully deleted")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのdeleteSignalCatalog」を参照してください。

`deleteVehicle`

次の例は、`deleteVehicle` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteVehicle(vecName: String) {
    val request = DeleteVehicleRequest {
        vehicleName = vecName
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        fleetwiseClient.deleteVehicle(request)
        println("Vehicle $vecName was deleted successfully.")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのdeleteVehicle」を参照してください。

`getDecoderManifest`

次の例は、`getDecoderManifest` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Waits for the specified model manifest to become active.
 *
 * @param decNameVal the name of the model manifest to wait for
 */
suspend fun waitForDecoderManifestActive(decNameVal: String) {
    var elapsedSeconds = 0
    var lastStatus: ManifestStatus = ManifestStatus.Draft

    print("# Elapsed: 0s | Status: DRAFT")
    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        while (true) {
            delay(1000)
            elapsedSeconds++
            if (elapsedSeconds % 5 == 0) {
                val request = GetDecoderManifestRequest {
                    name = decNameVal
                }

                val response = fleetwiseClient.getDecoderManifest(request)
                lastStatus = response.status ?: ManifestStatus.Draft

                when (lastStatus) {
                    ManifestStatus.Active -> {
                        print("\rElapsed: ${elapsedSeconds}s | Status: ACTIVE #\n")
                        return
                    }
```

```
                    ManifestStatus.Invalid -> {
                        print("\rElapsed: ${elapsedSeconds}s | Status: INVALID #\n")
                        throw RuntimeException("Model manifest became INVALID.
 Cannot proceed.")
                    }

                    else -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status:
 $lastStatus")
                    }
                }
            } else {
                print("\r Elapsed: ${elapsedSeconds}s | Status: $lastStatus")
            }
        }
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのgetDecoderManifest」を参照してください。

`getModelManifest`

次の例は、`getModelManifest` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Waits for the specified model manifest to become active.
 *
 * @param manifestName the name of the model manifest to wait for
 */
suspend fun waitForModelManifestActive(manifestNameVal: String) {
```

```
    var elapsedSeconds = 0
    var lastStatus: ManifestStatus = ManifestStatus.Draft

    print("# Elapsed: 0s | Status: DRAFT")
    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        while (true) {
            delay(1000)
            elapsedSeconds++
            if (elapsedSeconds % 5 == 0) {
                val request = GetModelManifestRequest {
                    name = manifestNameVal
                }

                val response = fleetwiseClient.getModelManifest(request)
                lastStatus = response.status ?: ManifestStatus.Draft

                when (lastStatus) {
                    ManifestStatus.Active -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status: ACTIVE #\n")
                        return
                    }

                    ManifestStatus.Invalid -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status: INVALID #
\n")
                        throw RuntimeException("Model manifest became INVALID.
 Cannot proceed.")
                    }

                    else -> {
                        print("\r Elapsed: ${elapsedSeconds}s | Status:
 $lastStatus")
                    }
                }
            } else {
                print("\r Elapsed: ${elapsedSeconds}s | Status: $lastStatus")
            }
        }
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのgetModelManifest」を参照してください。

`getVehicle`

次の例は、`getVehicle` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getVehicleDetails(vehicleNameVal: String) {
    val request = GetVehicleRequest {
        vehicleName = vehicleNameVal
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        val response = fleetwiseClient.getVehicle(request)
        val details = mapOf(
            "vehicleName" to response.vehicleName,
            "arn" to response.arn,
            "modelManifestArn" to response.modelManifestArn,
            "decoderManifestArn" to response.decoderManifestArn,
            "attributes" to response.attributes.toString(),
            "creationTime" to response.creationTime.toString(),
            "lastModificationTime" to response.lastModificationTime.toString(),
        )

        println("Vehicle Details:")
        for ((key, value) in details) {
            println("• %-20s : %s".format(key, value))
        }
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのgetVehicle」を参照してください。

`listSignalCatalogNodes`

次の例は、`listSignalCatalogNodes` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Lists the signal catalog nodes asynchronously.
 *
 * @param signalCatalogName the name of the signal catalog
 * @return a CompletableFuture that, when completed, contains a list of nodes in the
 specified signal catalog
 * @throws CompletionException if an exception occurs during the asynchronous
 operation
 */
suspend fun listSignalCatalogNode(signalCatalogName: String): List<Node>? {
    val request = ListSignalCatalogNodesRequest {
        name = signalCatalogName
    }

    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        val response = fleetwiseClient.listSignalCatalogNodes(request)
        return response.nodes
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのlistSignalCatalogNodes」を参照してください。

updateDecoderManifest

次の例は、updateDecoderManifest を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun updateDecoderManifest(nameVal: String) {
    val request = UpdateDecoderManifestRequest {
        name = nameVal
        status = ManifestStatus.Active
    }
    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        fleetwiseClient.updateDecoderManifest(request)
        println("$nameVal was successfully updated")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのupdateDecoderManifest」を参照してください。

updateModelManifest

次の例は、updateModelManifest を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
```

```
 * Updates the model manifest.
 *
 * @param nameVal the name of the model manifest to update
 */
suspend fun updateModelManifest(nameVal: String) {
    val request = UpdateModelManifestRequest {
        name = nameVal
        status = ManifestStatus.Active
    }
    IotFleetWiseClient { region = "us-east-1" }.use { fleetwiseClient ->
        fleetwiseClient.updateModelManifest(request)
        println("$nameVal was successfully updated")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのupdateModelManifest」を参照してください。

SDK for Kotlin を使用した Amazon Keyspaces の例

次のコード例は、Amazon Keyspaces で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
*/

suspend fun main() {
    listKeyspaces()
}

suspend fun listKeyspaces() {
    val keyspacesRequest =
        ListKeyspacesRequest {
            maxResults = 10
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.listKeyspaces(keyspacesRequest)
        response.keyspaces?.forEach { keyspace ->
            println("The name of the keyspace is ${keyspace.keyspaceName}")
        }
    }
}
```

- API の詳細については、『AWS SDK for Kotlin API リファレンス』の「ListKeyspaces」を参照してください。

トピック

- 基本
- アクション

基本

基本を学ぶ

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。テーブルスキーマにはムービーデータが保存され、特定時点のリカバリ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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:

 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This example uses a secure file format to hold certificate information for
 Kotlin 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 Kotlin 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.
 */

/*
   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.
 */
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main() {
    val fileName = "<Replace with the JSON file that contains movie data>"
    val keyspaceName = "<Replace with the name of the keyspace to create>"
    val titleUpdate = "The Family"
    val yearUpdate = 2013
    val tableName = "MovieKotlin"
    val tableNameRestore = "MovieRestore"

    val loader = DriverConfigLoader.fromClasspath("application.conf")
    val session =
        CqlSession
            .builder()
```

```
            .withConfigLoader(loader)
            .build()

    println(DASHES)
    println("Welcome to the Amazon Keyspaces example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create a keyspace.")
    createKeySpace(keyspaceName)
    println(DASHES)

    println(DASHES)
    delay(5000)
    println("2. Check for keyspace existence.")
    checkKeyspaceExistence(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("3. List keyspaces using a paginator.")
    listKeyspacesPaginator()
    println(DASHES)

    println(DASHES)
    println("4. Create a table with a simple movie data schema and enable point-in-
time recovery.")
    createTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("5. Check for the table to be in an Active state.")
    delay(6000)
    checkTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("6. List all tables in the keyspace.")
    listTables(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("7. Use a Cassandra driver to insert some records into the Movie
 table.")
    delay(6000)
```

```
    loadData(session, fileName, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("8. Get all records from the Movie table.")
    getMovieData(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("9. Get a specific Movie.")
    getSpecificMovie(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("10. Get a UTC timestamp for the current time.")
    val utc = ZonedDateTime.now(ZoneOffset.UTC)
    println("DATETIME = ${Date.from(utc.toInstant())}")
    println(DASHES)

    println(DASHES)
    println("11. Update the table schema to add a watched Boolean column.")
    updateTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("12. Update an item as watched.")
    delay(10000) // Wait 10 seconds for the update.
    updateRecord(session, keyspaceName, titleUpdate, yearUpdate)
    println(DASHES)

    println(DASHES)
    println("13. Query for items with watched = True.")
    getWatchedData(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("14. Restore the table back to the previous state using the timestamp.")
    println("Note that the restore operation can take up to 20 minutes.")
    restoreTable(keyspaceName, utc)
    println(DASHES)

    println(DASHES)
    println("15. Check for completion of the restore action.")
    delay(5000)
```

```
    checkRestoredTable(keyspaceName, "MovieRestore")
    println(DASHES)

    println(DASHES)
    println("16. Delete both tables.")
    deleteTable(keyspaceName, tableName)
    deleteTable(keyspaceName, tableNameRestore)
    println(DASHES)

    println(DASHES)
    println("17. Confirm that both tables are deleted.")
    checkTableDelete(keyspaceName, tableName)
    checkTableDelete(keyspaceName, tableNameRestore)
    println(DASHES)

    println(DASHES)
    println("18. Delete the keyspace.")
    deleteKeyspace(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("The scenario has completed successfully.")
    println(DASHES)
}

suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}

suspend fun checkTableDelete(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var status: String
    var response: GetTableResponse
    val tableRequest =
        GetTableRequest {
```

```
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    try {
        KeyspacesClient { region = "us-east-1" }.use { keyClient ->
            // Keep looping until the table cannot be found and a
 ResourceNotFoundException is thrown.
            while (true) {
                response = keyClient.getTable(tableRequest)
                status = response.status.toString()
                println(". The table status is $status")
                delay(500)
            }
        }
    } catch (e: ResourceNotFoundException) {
        println(e.message)
    }
    println("The table is deleted")
}

suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}

suspend fun checkRestoredTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null
```

```
    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println("The table status is $status")

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }

        val cols = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }
```

```
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}

fun getWatchedData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin\"
 WHERE watched = true ALLOW FILTERING;")
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

fun updateRecord(
    session: CqlSession,
    keySpace: String,
    titleUpdate: String?,
    yearUpdate: Int,
) {
    val sqlStatement =
        "UPDATE \"$keySpace\".\"MovieKotlin\" SET watched=true WHERE title = :k0 AND
 year = :k1;"
    val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
    builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
    val preparedStatement = session.prepare(sqlStatement)
    builder.addStatement(
        preparedStatement
            .boundStatementBuilder()
            .setString("k0", titleUpdate)
            .setInt("k1", yearUpdate)
            .build(),
    )
    val batchStatement = builder.build()
    session.execute(batchStatement)
}

suspend fun updateTable(
```

```
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}

fun getSpecificMovie(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet =
        session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin\" WHERE title
 = 'The Family' ALLOW FILTERING ;")

    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Get records from the Movie table.
fun getMovieData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin
\";")
    resultSet.forEach { item: Row ->
```

```
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Load data into the table.
fun loadData(
    session: CqlSession,
    fileName: String,
    keySpace: String,
) {
    val sqlStatement =
        "INSERT INTO \"$keySpace\".\"MovieKotlin\" (title, year, plot) values
 (:k0, :k1, :k2)"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        // Insert the data into the Amazon Keyspaces table.
        val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
        val preparedStatement: PreparedStatement = session.prepare(sqlStatement)
        builder.addStatement(
            preparedStatement
                .boundStatementBuilder()
                .setString("k0", title)
                .setInt("k1", year)
                .setString("k2", info)
                .build(),
        )
```

```
        val batchStatement = builder.build()
        session.execute(batchStatement)
        t++
    }
}

suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}

suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
```

```
        }
        val cols: List<ColumnDefinition>? = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
    colList.add(defYear)
    colList.add(defReleaseDate)
```

```
    colList.add(defPlot)

    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }

    val titleKey =
        PartitionKey {
            name = "title"
        }

    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)

    val schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}

suspend fun listKeyspacesPaginator() {
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
```

```
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}

suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse = keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}

suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- API の詳細については、『AWS SDK for Kotlin 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

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- API の詳細については、『AWS SDK for Kotlin API リファレンス』の「CreateKeyspace」を参照してください。

`CreateTable`

次の例は、`CreateTable` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
```

```
    colList.add(defYear)
    colList.add(defReleaseDate)
    colList.add(defPlot)

    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }

    val titleKey =
        PartitionKey {
            name = "title"
        }

    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)

    val schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateTable」を参照してください。

`DeleteKeyspace`

次の例は、`DeleteKeyspace` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}
```

- API の詳細については、『AWS SDK for Kotlin API リファレンス』の「DeleteKeyspace」を参照してください。

`DeleteTable`

次の例は、`DeleteTable` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}
```

- API の詳細については、『AWS SDK for Kotlin API リファレンス』の「DeleteTable」を参照してください。

`GetKeyspace`

次の例は、`GetKeyspace` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
```

```
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse = keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}
```

- API の詳細については、『AWS SDK for Kotlin API リファレンス』の「GetKeyspace」を参照してください。

`GetTable`

次の例は、`GetTable` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
```

```
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}
```

- API の詳細については、『AWS SDK for Kotlin API リファレンス』の「GetTable」を参照してください。

`ListKeyspaces`

次の例は、`ListKeyspaces` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listKeyspacesPaginator() {
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
```

```
                println("Name: ${obj.keyspaceName}")
            }
    }
}
```

- API の詳細については、『AWS SDK for Kotlin API リファレンス』の「ListKeyspaces」を参照してください。

`ListTables`

次の例は、`ListTables` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}
```

- API の詳細については、『AWS SDK for Kotlin API リファレンス』の「ListTables」を参照してください。

`RestoreTable`

次の例は、`RestoreTable` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}
```

- API の詳細については、『AWS SDK for Kotlin API リファレンス』の「RestoreTable」を参照してください。

`UpdateTable`

次の例は、UpdateTable を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}
```

- API の詳細については、『AWS SDK for Kotlin API リファレンス』の「UpdateTable」を参照してください。

AWS KMS SDK for Kotlin を使用した の例

次のコード例は、 で AWS SDK for Kotlin 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を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- アクション

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createCustomAlias(
    targetKeyIdVal: String?,
    aliasNameVal: String?,
) {
    val request =
        CreateAliasRequest {
            aliasName = aliasNameVal
            targetKeyId = targetKeyIdVal
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
```

```
        kmsClient.createAlias(request)
        println("$aliasNameVal was successfully created")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateAlias」を参照してください。

`CreateGrant`

次の例は、CreateGrant を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createNewGrant(
    keyIdVal: String?,
    granteePrincipalVal: String?,
    operation: String,
): String? {
    val operationOb = GrantOperation.fromValue(operation)
    val grantOperationList = ArrayList<GrantOperation>()
    grantOperationList.add(operationOb)

    val request =
        CreateGrantRequest {
            keyId = keyIdVal
            granteePrincipal = granteePrincipalVal
            operations = grantOperationList
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.createGrant(request)
        return response.grantId
    }
```

```
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CreateGrant」を参照してください。

`CreateKey`

次の例は、`CreateKey` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createKey(keyDesc: String?): String? {
    val request =
        CreateKeyRequest {
            description = keyDesc
            customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
            keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val result = kmsClient.createKey(request)
        println("Created a customer key with id " + result.keyMetadata?.arn)
        return result.keyMetadata?.keyId
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CreateKey」を参照してください。

`Decrypt`

次の例は、`Decrypt` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
```

```
        print(myVal)
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「Decrypt」を参照してください。

`DescribeKey`

次の例は、`DescribeKey` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeSpecifcKey(keyIdVal: String?) {
    val request =
        DescribeKeyRequest {
            keyId = keyIdVal
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.describeKey(request)
        println("The key description is ${response.keyMetadata?.description}")
        println("The key ARN is ${response.keyMetadata?.arn}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeKey」を参照してください。

`DisableKey`

次の例は、`DisableKey` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun disableKey(keyIdVal: String?) {
    val request =
        DisableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        kmsClient.disableKey(request)
        println("$keyIdVal was successfully disabled")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DisableKey」を参照してください。

`EnableKey`

次の例は、`EnableKey` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun enableKey(keyIdVal: String?) {
    val request =
        EnableKeyRequest {
```

```
            keyId = keyIdVal
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        kmsClient.enableKey(request)
        println("$keyIdVal was successfully enabled.")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「EnableKey」を参照してください。

`Encrypt`

次の例は、`Encrypt` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
```

```
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
        print(myVal)
    }
}
```

- API の詳細については、AWS SDK for Kotlin 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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listAllAliases() {
    val request =
        ListAliasesRequest {
```

```
            limit = 15
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listAliases(request)
        response.aliases?.forEach { alias ->
            println("The alias name is ${alias.aliasName}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ListAliases」を参照してください。

`ListGrants`

次の例は、`ListGrants` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun displayGrantIds(keyIdVal: String?) {
    val request =
        ListGrantsRequest {
            keyId = keyIdVal
            limit = 15
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listGrants(request)
        response.grants?.forEach { grant ->
            println("The grant Id is ${grant.grantId}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ListGrants」を参照してください。

`ListKeys`

次の例は、`ListKeys` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listAllKeys() {
    val request =
        ListKeysRequest {
            limit = 15
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listKeys(request)
        response.keys?.forEach { key ->
            println("The key ARN is ${key.keyArn}")
            println("The key Id is ${key.keyId}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ListKeys」を参照してください。

SDK for Kotlin を使用する Lambda の例

次のコード例は、Lambda で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、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、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

トピック

- 基本
- アクション
- シナリオ

基本

基本を学ぶ

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <functionName> <role> <handler> <bucketName> <updatedBucketName> <key>

        Where:
            functionName - The name of the AWS Lambda function.
            role - The AWS Identity and Access Management (IAM) service role that
 has AWS Lambda permissions.
            handler - The fully qualified method name (for example,
 example.Handler::handleRequest).
            bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name
 that contains the ZIP or JAR used for the Lambda function's code.
            updatedBucketName - The Amazon S3 bucket name that contains the .zip
 or .jar used to update the Lambda function's code.
            key - The Amazon S3 key name that represents the .zip or .jar file (for
 example, LambdaHello-1.0-SNAPSHOT.jar).
            """

   if (args.size != 6) {
        println(usage)
        exitProcess(1)
   }

   val functionName = args[0]
   val role = args[1]
   val handler = args[2]
   val bucketName = args[3]
   val updatedBucketName = args[4]
   val key = args[5]

   println("Creating a Lambda function named $functionName.")
   val funArn = createScFunction(functionName, bucketName, key, handler, role)
   println("The AWS Lambda ARN is $funArn")

   // Get a specific Lambda function.
   println("Getting the $functionName AWS Lambda function.")
   getFunction(functionName)

   // List the Lambda functions.
   println("Listing all AWS Lambda functions.")
   listFunctionsSc()
```

```
    // Invoke the Lambda function.
    println("*** Invoke the Lambda function.")
    invokeFunctionSc(functionName)

    // Update the AWS Lambda function code.
    println("*** Update the Lambda function code.")
    updateFunctionCode(functionName, updatedBucketName, key)

    // println("*** Invoke the function again after updating the code.")
    invokeFunctionSc(functionName)

    // Update the AWS Lambda function configuration.
    println("Update the run time of the function.")
    updateFunctionConfiguration(functionName, handler)

    // Delete the AWS Lambda function.
    println("Delete the AWS Lambda function.")
    delFunction(functionName)
}

suspend fun createScFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String,
): String {
    val functionCode =
        FunctionCode {
            s3Bucket = s3BucketName
            s3Key = myS3Key
        }

    val request =
        CreateFunctionRequest {
            functionName = myFunctionName
            code = functionCode
            description = "Created by the Lambda Kotlin API"
            handler = myHandler
            role = myRole
            runtime = Runtime.Java17
        }

    // Create a Lambda function using a waiter
```

```
    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn.toString()
    }
}

suspend fun getFunction(functionNameVal: String) {
    val functionRequest =
        GetFunctionRequest {
            functionName = functionNameVal
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.getFunction(functionRequest)
        println("The runtime of this Lambda function is
 ${response.configuration?.runtime}")
    }
}

suspend fun listFunctionsSc() {
    val request =
        ListFunctionsRequest {
            maxItems = 10
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.listFunctions(request)
        response.functions?.forEach { function ->
            println("The function name is ${function.functionName}")
        }
    }
}

suspend fun invokeFunctionSc(functionNameVal: String) {
    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request =
        InvokeRequest {
            functionName = functionNameVal
            payload = byteArray
            logType = LogType.Tail
```

```
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("The function payload is ${res.payload?.toString(Charsets.UTF_8)}")
    }
}

suspend fun updateFunctionCode(
    functionNameVal: String?,
    bucketName: String?,
    key: String?,
) {
    val functionCodeRequest =
        UpdateFunctionCodeRequest {
            functionName = functionNameVal
            publish = true
            s3Bucket = bucketName
            s3Key = key
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.updateFunctionCode(functionCodeRequest)
        awsLambda.waitUntilFunctionUpdated {
            functionName = functionNameVal
        }
        println("The last modified value is " + response.lastModified)
    }
}

suspend fun updateFunctionConfiguration(
    functionNameVal: String?,
    handlerVal: String?,
) {
    val configurationRequest =
        UpdateFunctionConfigurationRequest {
            functionName = functionNameVal
            handler = handlerVal
            runtime = Runtime.Java17
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.updateFunctionConfiguration(configurationRequest)
    }
```

```
}

suspend fun delFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin 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

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createNewFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String,
): String? {
    val functionCode =
        FunctionCode {
            s3Bucket = s3BucketName
            s3Key = myS3Key
        }

    val request =
        CreateFunctionRequest {
            functionName = myFunctionName
            code = functionCode
            description = "Created by the Lambda Kotlin API"
            handler = myHandler
            role = myRole
            runtime = Runtime.Java17
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateFunction」を参照してください。

`DeleteFunction`

次の例は、`DeleteFunction` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun delLambdaFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteFunction」を参照してください。

`Invoke`

次の例は、`Invoke` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun invokeFunction(functionNameVal: String) {
    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request =
```

```
        InvokeRequest {
            functionName = functionNameVal
            logType = LogType.Tail
            payload = byteArray
        }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("${res.payload?.toString(Charsets.UTF_8)}")
        println("The log result is ${res.logResult}")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「Invoke」を参照してください。

シナリオ

サーバーレスアプリケーションを作成して写真の管理

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

SDK for Kotlin

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設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、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

SDK for Kotlin を使用した Amazon Location の例

次のコード例は、Amazon Location で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

こんにちは Amazon Location Service

次のコード例は、Amazon Location Service の使用を開始する方法を示しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

In addition, you need to create a collection using the AWS Management
console. For information, see the following documentation.

https://docs.aws.amazon.com/location/latest/developerguide/geofence-gs.html
```

```
 */
suspend fun main(args: Array<String>) {
    val usage = """

        Usage:
            <colletionName>

        Where:
            colletionName - The Amazon location collection name.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }
    val colletionName = args[0]
    listGeofences(colletionName)
}

/**
 * Lists the geofences for the specified collection name.
 *
 * @param collectionName the name of the geofence collection
 */
suspend fun listGeofences(collectionName: String) {
    val request = ListGeofencesRequest {
        this.collectionName = collectionName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        val response = client.listGeofences(request)
        val geofences = response.entries
        if (geofences.isNullOrEmpty()) {
            println("No Geofences found")
        } else {
            geofences.forEach { geofence ->
                println("Geofence ID: ${geofence.geofenceId}")
            }
        }
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのListGeofencesPaginator」を参照してください。

トピック

- 基本
- アクション

基本

基本を学ぶ

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 Location マップを作成します。
- Amazon Location API キーを作成します。
- マップ URL を表示します。
- ジオフェンスコレクションを作成します。
- ジオフェンスジオメトリを保存します。
- トラッカーリソースを作成します。
- デバイスの位置を更新します。
- 指定されたデバイスの最新の位置更新を取得します。
- ルート計算ツールを作成します。
- シアトルとバンクーバー間の距離を決定します。
- Amazon Location の上位レベルの APIs。
- Amazon Location Assets を削除します。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

val scanner = Scanner(System.`in`)
val DASHES = String(CharArray(80)).replace("\u0000", "-")
suspend fun main(args: Array<String>) {
    val usage = """

        Usage:    <mapName> <keyName> <collectionName> <geoId> <trackerName>
 <calculatorName> <deviceId>

        Where:
            mapName - The name of the map to create (e.g., "AWSMap").
            keyName - The name of the API key to create (e.g., "AWSApiKey").
            collectionName - The name of the geofence collection (e.g.,
 "AWSLocationCollection").
            geoId - The geographic identifier used for the geofence or map (e.g.,
 "geoId").
            trackerName - The name of the tracker (e.g., "geoTracker").
            calculatorName - The name of the route calculator (e.g.,
 "AWSRouteCalc").
            deviceId - The ID of the device (e.g., "iPhone-112356").
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(0)
    }

    val mapName = args[0]
    val keyName = args[1]
    val collectionName = args[2]
    val geoId = args[3]
    val trackerName = args[4]
    val calculatorName = args[5]
    val deviceId = args[6]

    println(
```

```
        """
    AWS Location Service is a fully managed service offered by Amazon Web Services
(AWS) that
    provides location-based services for developers. This service simplifies
    the integration of location-based features into applications, making it
    easier to build and deploy location-aware applications.

    The AWS Location Service offers a range of location-based services,
    including:

    - Maps: The service provides access to high-quality maps, satellite imagery,
      and geospatial data from various providers, allowing developers to
      easily embed maps into their applications.

    - Tracking: The Location Service enables real-time tracking of mobile devices,
      assets, or other entities, allowing developers to build applications
      that can monitor the location of people, vehicles, or other objects.

    - Geocoding: The service provides the ability to convert addresses or
      location names into geographic coordinates (latitude and longitude),
      and vice versa, enabling developers to integrate location-based search
      and routing functionality into their applications.
        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    println(DASHES)
    println("1. Create an AWS Location Service map")
    println(
        """
        An AWS Location map can enhance the user experience of your
        application by providing accurate and personalized location-based
        features. For example, you could use the geocoding capabilities to
        allow users to search for and locate businesses, landmarks, or
        other points of interest within a specific region.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    val mapArn = createMap(mapName)
    println("The Map ARN is: $mapArn")
    waitForInputToContinue(scanner)
    println(DASHES)
```

```
    waitForInputToContinue(scanner)
    println("2. Create an AWS Location API key")
    println(
        """
            When you embed a map in a web app or website, the API key is
            included in the map tile URL to authenticate requests. You can
            restrict API keys to specific AWS Location operations (e.g., only
            maps, not geocoding). API keys can expire, ensuring temporary
            access control.

        """.trimIndent(),
    )
    val keyArn = createKey(keyName, mapArn)
    println("The Key ARN is: $keyArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("3. Display Map URL")
    println(
        """
        In order to get the MAP URL, you need to get the API Key value.
        You can get the key value using the AWS Management Console under
        Location Services. This operation cannot be completed using the
        AWS SDK. For more information about getting the key value, see
        the AWS Location Documentation.
        """.trimIndent(),
    )
    val mapUrl = "https://maps.geo.aws.amazon.com/maps/v0/maps/$mapName/tiles/{z}/
{x}/{y}?key={KeyValue}"
    println("Embed this URL in your Web app: $mapUrl")
    println("")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("4. Create a geofence collection, which manages and stores geofences.")
    waitForInputToContinue(scanner)
    val collectionArn: String =
        createGeofenceCollection(collectionName)
    println("The geofence collection was successfully created: $collectionArn")
    waitForInputToContinue(scanner)
```

```
    println(DASHES)
    println("5. Store a geofence geometry in a given geofence collection.")
    println(
        """
        An AWS Location geofence is a virtual boundary that defines a geographic
area
        on a map. It is a useful feature for tracking the location of
        assets or monitoring the movement of objects within a specific region.

        To define a geofence, you need to specify the coordinates of a
        polygon that represents the area of interest. The polygon must be
        defined in a counter-clockwise direction, meaning that the points of
        the polygon must be listed in a counter-clockwise order.

        This is a requirement for the AWS Location service to correctly
        interpret the geofence and ensure that the location data is
        accurately processed within the defined area.
        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    putGeofence(collectionName, geoId)
    println("Successfully created geofence: $geoId")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("6. Create a tracker resource which lets you retrieve current and
historical location of devices.")
    waitForInputToContinue(scanner)
    val trackerArn: String = createTracker(trackerName)
    println("Successfully created tracker. ARN: $trackerArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("7. Update the position of a device in the location tracking system.")
    println(
        """
        The AWS location service does not enforce a strict format for deviceId, but
it must:
            - Be a string (case-sensitive).
            - Be 1–100 characters long.
            - Contain only:
```

```
            - Alphanumeric characters (A-Z, a-z, 0-9)
            - Underscores (_)
            - Hyphens (-)
            - Be the same ID used when sending and retrieving positions.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    updateDevicePosition(trackerName, deviceId)
    println("$deviceId was successfully updated in the location tracking system.")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("8. Retrieve the most recent position update for a specified device.")
    waitForInputToContinue(scanner)
    val response = getDevicePosition(trackerName, deviceId)
    println("Successfully fetched device position: ${response.position}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("9. Create a route calculator.")
    waitForInputToContinue(scanner)
    val routeResponse = createRouteCalculator(calculatorName)
    println("Route calculator created successfully: ${routeResponse.calculatorArn}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("10. Determine the distance in kilometers between Seattle and Vancouver
using the route calculator.")
    waitForInputToContinue(scanner)
    val responseDis = calcDistance(calculatorName)
    println("Successfully calculated route. The distance in kilometers is
${responseDis.summary?.distance}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("11. Use the GeoPlacesClient to perform additional operations.")
    println(
        """
```

```
        This scenario will show use of the GeoPlacesClient that enables
        location search and geocoding capabilities for your applications.

        We are going to use this client to perform these AWS Location tasks:
            - Reverse Geocoding (reverseGeocode): Converts geographic coordinates
into addresses.
            - Place Search (searchText): Finds places based on search queries.
            - Nearby Search (searchNearby): Finds places near a specific location.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    println("First we will perform a Reverse Geocoding operation")
    waitForInputToContinue(scanner)
    reverseGeocode()

    println("Now we are going to perform a text search using coffee shop.")
    waitForInputToContinue(scanner)
    searchText("coffee shop")
    waitForInputToContinue(scanner)

    println("Now we are going to perform a nearby Search.")
    waitForInputToContinue(scanner)
    searchNearby()
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS Location Services resources.")
    println("Would you like to delete the AWS Location Services resources? (y/n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
        deleteMap(mapName)
        deleteKey(keyName)
        deleteGeofenceCollection(collectionName)
        deleteTracker(trackerName)
        deleteRouteCalculator(calculatorName)
    } else {
        println("The AWS resources will not be deleted.")
    }
    waitForInputToContinue(scanner)
    println(DASHES)
```

```
    println(DASHES)
    println(" This concludes the AWS Location Service scenario.")
    println(DASHES)
}

/**
 * Deletes a route calculator from the system.
 * @param calcName the name of the route calculator to delete
 */
suspend fun deleteRouteCalculator(calcName: String) {
    val calculatorRequest = DeleteRouteCalculatorRequest {
        this.calculatorName = calcName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteRouteCalculator(calculatorRequest)
        println("The route calculator $calcName was deleted.")
    }
}


/**
 * Deletes a tracker with the specified name.
 * @param trackerName the name of the tracker to be deleted
 */
suspend fun deleteTracker(trackerName: String) {
    val trackerRequest = DeleteTrackerRequest {
        this.trackerName = trackerName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteTracker(trackerRequest)
        println("The tracker $trackerName was deleted.")
    }
}


/**
 * Deletes a geofence collection.
 *
 * @param collectionName the name of the geofence collection to be deleted
 * @return a {@link CompletableFuture} that completes when the geofence collection
 has been deleted
 */
```

```
suspend fun deleteGeofenceCollection(collectionName: String) {
    val collectionRequest = DeleteGeofenceCollectionRequest {
        this.collectionName = collectionName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteGeofenceCollection(collectionRequest)
        println("The geofence collection $collectionName was deleted.")
    }
}

/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteKey(keyName: String) {
    val keyRequest = DeleteKeyRequest {
        this.keyName = keyName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteKey(keyRequest)
        println("The key $keyName was deleted.")
    }
}

/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteMap(mapName: String) {
    val mapRequest = DeleteMapRequest {
        this.mapName = mapName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteMap(mapRequest)
        println("The map $mapName was deleted.")
    }
}
```

```
/**
 * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
 * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
 * The results are processed and printed once the search completes successfully.
 */
suspend fun searchNearby() {
    val latitude = 37.7749
    val longitude = -122.4194
    val queryPosition = listOf(longitude, latitude)

    // Set up the request for searching nearby places.
    val request = SearchNearbyRequest {
        this.queryPosition = queryPosition
        this.queryRadius = 1000L
    }

    GeoPlacesClient { region = "us-east-1" }.use { client ->
        val response = client.searchNearby(request)

        // Process the response and print the results.
        response.resultItems?.forEach { result ->
            println("Title: ${result.title}")
            println("Address: ${result.address?.label}")
            println("Distance: ${result.distance} meters")
            println("-------------------------")
        }
    }
}


/**
 * Searches for a place using the provided search query and prints the detailed
 information of the first result.
 *
 * @param searchQuery the search query to be used for the place search (ex, coffee
 shop)
 */
suspend fun searchText(searchQuery: String) {
    val latitude = 37.7749
    val longitude = -122.4194
    val queryPosition = listOf(longitude, latitude)
```

```
    val request = SearchTextRequest {
        this.queryText = searchQuery
        this.biasPosition = queryPosition
    }

    GeoPlacesClient { region = "us-east-1" }.use { client ->
        val response = client.searchText(request)

        response.resultItems?.firstOrNull()?.let { result ->
            val placeId = result.placeId // Get Place ID
            println("Found Place with id: $placeId")

            // Fetch detailed info using getPlace.
            val getPlaceRequest = GetPlaceRequest {
                this.placeId = placeId
            }

            val placeResponse = client.getPlace(getPlaceRequest)

            // Print detailed place information.
            println("Detailed Place Information:")
            println("Title: ${placeResponse.title}")
            println("Address: ${placeResponse.address?.label}")

            // Print each food type (if any).
            placeResponse.foodTypes?.takeIf { it.isNotEmpty() }?.let {
                println("Food Types:")
                it.forEach { foodType ->
                    println("  - $foodType")
                }
            } ?: run {
                println("No food types available.")
            }

            println("-------------------------")
        }
    }
}

/**
 * Performs reverse geocoding using the AWS Geo Places API.
 * Reverse geocoding is the process of converting geographic coordinates (latitude
 and longitude) to a human-readable address.
```

```
 * This method uses the latitude and longitude of San Francisco as the input, and
 prints the resulting address.
 */
suspend fun reverseGeocode() {
    val latitude = 37.7749
    val longitude = -122.4194
    println("Use latitude 37.7749 and longitude -122.4194")

    // AWS expects [longitude, latitude].
    val queryPosition = listOf(longitude, latitude)
    val request = ReverseGeocodeRequest {
        this.queryPosition = queryPosition
    }

    GeoPlacesClient { region = "us-east-1" }.use { client ->
        val response = client.reverseGeocode(request)
        response.resultItems?.forEach { result ->
            println("The address is: ${result.address?.label}")
        }
    }
}


/**
 * Calculates the distance between two locations.
 *
 * @param routeCalcName the name of the route calculator to use
 * @return a {@link CompletableFuture} that will complete with a {@link
 CalculateRouteResponse} containing the distance and estimated duration of the route
 */
suspend fun calcDistance(routeCalcName: String): CalculateRouteResponse {
    // Define coordinates for Seattle, WA and Vancouver, BC.
    val departurePosition = listOf(-122.3321, 47.6062)
    val arrivePosition = listOf(-123.1216, 49.2827)

    val request = CalculateRouteRequest {
        this.calculatorName = routeCalcName
        this.departurePosition = departurePosition
        this.destinationPosition = arrivePosition
        this.travelMode = TravelMode.Car // Options: Car, Truck, Walking, Bicycle
        this.distanceUnit = DistanceUnit.Kilometers // Options: Meters, Kilometers,
 Miles
    }
```

```
    LocationClient { region = "us-east-1" }.use { client ->
        return client.calculateRoute(request)
    }
}

/**
 * Creates a new route calculator with the specified name and data source.
 *
 * @param routeCalcName the name of the route calculator to be created
 */
suspend fun createRouteCalculator(routeCalcName: String):
 CreateRouteCalculatorResponse {
    val dataSource = "Esri"

    val request = CreateRouteCalculatorRequest {
        this.calculatorName = routeCalcName
        this.dataSource = dataSource
    }

    LocationClient { region = "us-east-1" }.use { client ->
        return client.createRouteCalculator(request)
    }
}

/**
 * Retrieves the position of a device using the provided LocationClient.
 *
 * @param trackerName The name of the tracker associated with the device.
 * @param deviceId    The ID of the device to retrieve the position for.
 */
suspend fun getDevicePosition(trackerName: String, deviceId: String):
 GetDevicePositionResponse {
    val request = GetDevicePositionRequest {
        this.trackerName = trackerName
        this.deviceId = deviceId
    }

    LocationClient { region = "us-east-1" }.use { client ->
        return client.getDevicePosition(request)
    }
}

/**
 * Updates the position of a device in the location tracking system.
```

```
 *
 * @param trackerName the name of the tracker associated with the device
 * @param deviceId    the unique identifier of the device
 */
suspend fun updateDevicePosition(trackerName: String, deviceId: String) {
    val latitude = 37.7749
    val longitude = -122.4194

    val positionUpdate = DevicePositionUpdate {
        this.deviceId = deviceId
        sampleTime = aws.smithy.kotlin.runtime.time.Instant.now() // Timestamp of
 position update.
        position = listOf(longitude, latitude) // AWS requires [longitude, latitude]
    }

    val request = BatchUpdateDevicePositionRequest {
        this.trackerName = trackerName
        updates = listOf(positionUpdate)
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.batchUpdateDevicePosition(request)
    }
}

/**
 * Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
 *
 * @param trackerName the name of the tracker to be created
 * @return a {@link CompletableFuture} that, when completed, will contain the Amazon
 Resource Name (ARN) of the created tracker
 */
suspend fun createTracker(trackerName: String): String {
    val trackerRequest = CreateTrackerRequest {
        description = "Created using the Kotlin SDK"
        this.trackerName = trackerName
        positionFiltering = PositionFiltering.TimeBased // Options: TimeBased,
 DistanceBased, AccuracyBased
    }

    LocationClient { region = "us-east-1" }.use { client ->
        val response = client.createTracker(trackerRequest)
        return response.trackerArn
```

```
    }
}

/**
 * Adds a new geofence to the specified collection.
 *
 * @param collectionName the name of the geofence collection to add the geofence to
 * @param geoId          the unique identifier for the geofence
 */
suspend fun putGeofence(collectionName: String, geoId: String) {
    val geofenceGeometry = GeofenceGeometry {
        polygon = listOf(
            listOf(
                listOf(-122.3381, 47.6101),
                listOf(-122.3281, 47.6101),
                listOf(-122.3281, 47.6201),
                listOf(-122.3381, 47.6201),
                listOf(-122.3381, 47.6101),
            ),
        )
    }

    val geofenceRequest = PutGeofenceRequest {
        this.collectionName = collectionName
        this.geofenceId = geoId
        this.geometry = geofenceGeometry
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.putGeofence(geofenceRequest)
    }
}

/**
 * Creates a new geofence collection.
 *
 * @param collectionName the name of the geofence collection to be created
 */
suspend fun createGeofenceCollection(collectionName: String): String {
    val collectionRequest = CreateGeofenceCollectionRequest {
        this.collectionName = collectionName
        description = "Created by using the AWS SDK for Kotlin"
    }
```

```
    LocationClient { region = "us-east-1" }.use { client ->
        val response = client.createGeofenceCollection(collectionRequest)
        return response.collectionArn
    }
}

/**
 * Creates a new API key with the specified name and restrictions.
 *
 * @param keyName the name of the API key to be created
 * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which the
 API key will be associated
 * @return the Amazon Resource Name (ARN) of the created API key
 */
suspend fun createKey(keyName: String, mapArn: String): String {
    val keyRestrictions = ApiKeyRestrictions {
        allowActions = listOf("geo:GetMap*")
        allowResources = listOf(mapArn)
    }

    val request = CreateKeyRequest {
        this.keyName = keyName
        this.restrictions = keyRestrictions
        noExpiry = true
    }

    LocationClient { region = "us-east-1" }.use { client ->
        val response = client.createKey(request)
        return response.keyArn
    }
}

/**
 * Creates a new map with the specified name and configuration.
 *
 * @param mapName the name of the map to be created
 * @return he Amazon Resource Name (ARN) of the created map
 */
suspend fun createMap(mapName: String): String {
    val configuration = MapConfiguration {
        style = "VectorEsriNavigation"
    }

    val mapRequest = CreateMapRequest {
```

```
        this.mapName = mapName
        this.configuration = configuration
        description = "A map created using the Kotlin SDK"
    }

    LocationClient { region = "us-east-1" }.use { client ->
        val response = client.createMap(mapRequest)
        return response.mapArn
    }
}

fun waitForInputToContinue(scanner: Scanner) {
    while (true) {
        println("")
        println("Enter 'c' followed by <ENTER> to continue:")
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else {
            println("Invalid input. Please try again.")
        }
    }
}
```

アクション

`BatchUpdateDevicePosition`

次の例は、`BatchUpdateDevicePosition` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Updates the position of a device in the location tracking system.
 *
 * @param trackerName the name of the tracker associated with the device
 * @param deviceId    the unique identifier of the device
 */
suspend fun updateDevicePosition(trackerName: String, deviceId: String) {
    val latitude = 37.7749
    val longitude = -122.4194

    val positionUpdate = DevicePositionUpdate {
        this.deviceId = deviceId
        sampleTime = aws.smithy.kotlin.runtime.time.Instant.now() // Timestamp of
 position update.
        position = listOf(longitude, latitude) // AWS requires [longitude, latitude]
    }

    val request = BatchUpdateDevicePositionRequest {
        this.trackerName = trackerName
        updates = listOf(positionUpdate)
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.batchUpdateDevicePosition(request)
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのBatchUpdateDevicePosition」を参照してください。

`CalculateRoute`

次の例は、`CalculateRoute` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Calculates the distance between two locations.
 *
 * @param routeCalcName the name of the route calculator to use
 * @return a {@link CompletableFuture} that will complete with a {@link
 CalculateRouteResponse} containing the distance and estimated duration of the route
 */
suspend fun calcDistance(routeCalcName: String): CalculateRouteResponse {
    // Define coordinates for Seattle, WA and Vancouver, BC.
    val departurePosition = listOf(-122.3321, 47.6062)
    val arrivePosition = listOf(-123.1216, 49.2827)

    val request = CalculateRouteRequest {
        this.calculatorName = routeCalcName
        this.departurePosition = departurePosition
        this.destinationPosition = arrivePosition
        this.travelMode = TravelMode.Car // Options: Car, Truck, Walking, Bicycle
        this.distanceUnit = DistanceUnit.Kilometers // Options: Meters, Kilometers,
 Miles
    }

    LocationClient { region = "us-east-1" }.use { client ->
        return client.calculateRoute(request)
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのCalculateRoute」を参照してください。

`CreateGeofenceCollection`

次の例は、`CreateGeofenceCollection` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Creates a new geofence collection.
 *
 * @param collectionName the name of the geofence collection to be created
 */
suspend fun createGeofenceCollection(collectionName: String): String {
    val collectionRequest = CreateGeofenceCollectionRequest {
        this.collectionName = collectionName
        description = "Created by using the AWS SDK for Kotlin"
    }

    LocationClient { region = "us-east-1" }.use { client ->
        val response = client.createGeofenceCollection(collectionRequest)
        return response.collectionArn
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスの「CreateGeofenceCollection」を参照してください。

`CreateKey`

次の例は、`CreateKey` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Creates a new API key with the specified name and restrictions.
 *
 * @param keyName the name of the API key to be created
 * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which the
 API key will be associated
 * @return the Amazon Resource Name (ARN) of the created API key
 */
```

```
suspend fun createKey(keyName: String, mapArn: String): String {
    val keyRestrictions = ApiKeyRestrictions {
        allowActions = listOf("geo:GetMap*")
        allowResources = listOf(mapArn)
    }

    val request = CreateKeyRequest {
        this.keyName = keyName
        this.restrictions = keyRestrictions
        noExpiry = true
    }

    LocationClient { region = "us-east-1" }.use { client ->
        val response = client.createKey(request)
        return response.keyArn
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CreateKey」を参照してください。

`CreateMap`

次の例は、`CreateMap` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Creates a new map with the specified name and configuration.
 *
 * @param mapName the name of the map to be created
 * @return he Amazon Resource Name (ARN) of the created map
 */
suspend fun createMap(mapName: String): String {
```

```
    val configuration = MapConfiguration {
        style = "VectorEsriNavigation"
    }

    val mapRequest = CreateMapRequest {
        this.mapName = mapName
        this.configuration = configuration
        description = "A map created using the Kotlin SDK"
    }

    LocationClient { region = "us-east-1" }.use { client ->
        val response = client.createMap(mapRequest)
        return response.mapArn
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスの「CreateMap」を参照してください。

`CreateRouteCalculator`

次の例は、`CreateRouteCalculator` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Creates a new route calculator with the specified name and data source.
 *
 * @param routeCalcName the name of the route calculator to be created
 */
suspend fun createRouteCalculator(routeCalcName: String):
 CreateRouteCalculatorResponse {
    val dataSource = "Esri"
```

```
    val request = CreateRouteCalculatorRequest {
        this.calculatorName = routeCalcName
        this.dataSource = dataSource
    }

    LocationClient { region = "us-east-1" }.use { client ->
        return client.createRouteCalculator(request)
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスの「CreateRouteCalculator」を参照してください。

`CreateTracker`

次の例は、`CreateTracker` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
 *
 * @param trackerName the name of the tracker to be created
 * @return a {@link CompletableFuture} that, when completed, will contain the Amazon
 Resource Name (ARN) of the created tracker
 */
suspend fun createTracker(trackerName: String): String {
    val trackerRequest = CreateTrackerRequest {
        description = "Created using the Kotlin SDK"
        this.trackerName = trackerName
        positionFiltering = PositionFiltering.TimeBased // Options: TimeBased,
 DistanceBased, AccuracyBased
    }
```

```
    LocationClient { region = "us-east-1" }.use { client ->
        val response = client.createTracker(trackerRequest)
        return response.trackerArn
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスの「CreateTracker」を参照してください。

`DeleteGeofenceCollection`

次の例は、`DeleteGeofenceCollection` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Deletes a geofence collection.
 *
 * @param collectionName the name of the geofence collection to be deleted
 * @return a {@link CompletableFuture} that completes when the geofence collection
 has been deleted
 */
suspend fun deleteGeofenceCollection(collectionName: String) {
    val collectionRequest = DeleteGeofenceCollectionRequest {
        this.collectionName = collectionName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteGeofenceCollection(collectionRequest)
        println("The geofence collection $collectionName was deleted.")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのDeleteGeofenceCollection」を参照してください。

`DeleteKey`

次の例は、DeleteKey を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteKey(keyName: String) {
    val keyRequest = DeleteKeyRequest {
        this.keyName = keyName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteKey(keyRequest)
        println("The key $keyName was deleted.")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのDeleteKey」を参照してください。

`DeleteMap`

次の例は、DeleteMap を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteMap(mapName: String) {
    val mapRequest = DeleteMapRequest {
        this.mapName = mapName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteMap(mapRequest)
        println("The map $mapName was deleted.")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのDeleteMap」を参照してください。

`DeleteRouteCalculator`

次の例は、`DeleteRouteCalculator` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Deletes a route calculator from the system.
 * @param calcName the name of the route calculator to delete
 */
suspend fun deleteRouteCalculator(calcName: String) {
    val calculatorRequest = DeleteRouteCalculatorRequest {
        this.calculatorName = calcName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteRouteCalculator(calculatorRequest)
        println("The route calculator $calcName was deleted.")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのDeleteRouteCalculator」を参照してください。

`DeleteTracker`

次の例は、`DeleteTracker` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Deletes a tracker with the specified name.
 * @param trackerName the name of the tracker to be deleted
 */
suspend fun deleteTracker(trackerName: String) {
    val trackerRequest = DeleteTrackerRequest {
        this.trackerName = trackerName
    }
```

```
    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteTracker(trackerRequest)
        println("The tracker $trackerName was deleted.")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのDeleteTracker」を参照してください。

`GetDevicePosition`

次の例は、`GetDevicePosition` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Retrieves the position of a device using the provided LocationClient.
 *
 * @param trackerName The name of the tracker associated with the device.
 * @param deviceId    The ID of the device to retrieve the position for.
 */
suspend fun getDevicePosition(trackerName: String, deviceId: String):
 GetDevicePositionResponse {
    val request = GetDevicePositionRequest {
        this.trackerName = trackerName
        this.deviceId = deviceId
    }

    LocationClient { region = "us-east-1" }.use { client ->
        return client.getDevicePosition(request)
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのGetDevicePosition」を参照してください。

`PutGeofence`

次の例は、PutGeofence を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Adds a new geofence to the specified collection.
 *
 * @param collectionName the name of the geofence collection to add the geofence to
 * @param geoId          the unique identifier for the geofence
 */
suspend fun putGeofence(collectionName: String, geoId: String) {
    val geofenceGeometry = GeofenceGeometry {
        polygon = listOf(
            listOf(
                listOf(-122.3381, 47.6101),
                listOf(-122.3281, 47.6101),
                listOf(-122.3281, 47.6201),
                listOf(-122.3381, 47.6201),
                listOf(-122.3381, 47.6101),
            ),
        )
    }

    val geofenceRequest = PutGeofenceRequest {
        this.collectionName = collectionName
        this.geofenceId = geoId
        this.geometry = geofenceGeometry
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.putGeofence(geofenceRequest)
```

```
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスのPutGeofence」を参照してください。

SDK for Kotlin を使用した MediaConvert の例

次のコード例は、MediaConvert で AWS SDK for Kotlin 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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createMediaJob(
    mcClient: MediaConvertClient,
    mcRoleARN: String,
```

```
    fileInputVal: String,
): String? {
    val s3path = fileInputVal.substring(0, fileInputVal.lastIndexOf('/') + 1) +
 "javasdk/out/"
    val fileOutput = s3path + "index"
    val thumbsOutput = s3path + "thumbs/"
    val mp4Output = s3path + "mp4/"

    try {
        val describeEndpoints =
            DescribeEndpointsRequest {
                maxResults = 20
            }

        val res = mcClient.describeEndpoints(describeEndpoints)
        if (res.endpoints?.size!! <= 0) {
            println("Cannot find MediaConvert service endpoint URL!")
            exitProcess(0)
        }
        val endpointURL = res.endpoints!!.get(0).url!!
        val mediaConvert =
            MediaConvertClient.fromEnvironment {
                region = "us-west-2"
                endpointProvider =
                    MediaConvertEndpointProvider {
                        Endpoint(endpointURL)
                    }
            }

        // output group Preset HLS low profile
        val hlsLow = createOutput("_low", "_\$dt$", 750000, 7, 1920, 1080, 640)

        // output group Preset HLS medium profile
        val hlsMedium = createOutput("_medium", "_\$dt$", 1200000, 7, 1920, 1080,
 1280)

        // output group Preset HLS high profole
        val hlsHigh = createOutput("_high", "_\$dt$", 3500000, 8, 1920, 1080, 1920)

        val outputSettings =
            OutputGroupSettings {
                type = OutputGroupType.HlsGroupSettings
            }
```

```
        val outputObsList: MutableList<Output> = mutableListOf()
        if (hlsLow != null) {
            outputObsList.add(hlsLow)
        }
        if (hlsMedium != null) {
            outputObsList.add(hlsMedium)
        }
        if (hlsHigh != null) {
            outputObsList.add(hlsHigh)
        }

        // Create an OutputGroup object.
        val appleHLS =
            OutputGroup {
                name = "Apple HLS"
                customName = "Example"
                outputGroupSettings =
                    OutputGroupSettings {
                        type = OutputGroupType.HlsGroupSettings
                        this.hlsGroupSettings =
                            HlsGroupSettings {
                                directoryStructure =
HlsDirectoryStructure.SingleDirectory
                                manifestDurationFormat =
HlsManifestDurationFormat.Integer
                                streamInfResolution = HlsStreamInfResolution.Include
                                clientCache = HlsClientCache.Enabled
                                captionLanguageSetting =
HlsCaptionLanguageSetting.Omit
                                manifestCompression = HlsManifestCompression.None
                                codecSpecification = HlsCodecSpecification.Rfc4281
                                outputSelection =
HlsOutputSelection.ManifestsAndSegments
                                programDateTime = HlsProgramDateTime.Exclude
                                programDateTimePeriod = 600
                                timedMetadataId3Frame =
HlsTimedMetadataId3Frame.Priv
                                timedMetadataId3Period = 10
                                destination = fileOutput
                                segmentControl = HlsSegmentControl.SegmentedFiles
                                minFinalSegmentLength = 0.toDouble()
                                segmentLength = 4
                                minSegmentLength = 1
                            }
```

```
                }
            outputs = outputObsList
        }

    val theOutput =
        Output {
            extension = "mp4"
            containerSettings =
                ContainerSettings {
                    container = ContainerType.fromValue("MP4")
                }

            videoDescription =
                VideoDescription {
                    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 {
                            codec = VideoCodec.H264
                            h264Settings =
                                H264Settings {
                                    rateControlMode = H264RateControlMode.Qvbr
                                    parControl =
H264ParControl.InitializeFromSource
                                    qualityTuningLevel =
H264QualityTuningLevel.SinglePass
                                    qvbrSettings = H264QvbrSettings
{ qvbrQualityLevel = 8 }
                                    codecLevel = H264CodecLevel.Auto
                                    codecProfile = H264CodecProfile.Main
                                    maxBitrate = 2400000
                                    framerateControl =
H264FramerateControl.InitializeFromSource
                                    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.DuplicateDrop
                                    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
                                }
                            }
                        }

                audioDescriptions =
                    listOf(
                        AudioDescription {
                            audioTypeControl = AudioTypeControl.FollowInput
                            languageCodeControl =
AudioLanguageCodeControl.FollowInput
                            codecSettings =
                                AudioCodecSettings {
                                    codec = AudioCodec.Aac
                                    aacSettings =
                                        AacSettings {
```

```
                                                codecProfile = AacCodecProfile.Lc
                                                rateControlMode = AacRateControlMode.Cbr
                                                codingMode = AacCodingMode.CodingMode2_0
                                                sampleRate = 44100
                                                bitrate = 160000
                                                rawFormat = AacRawFormat.None
                                                specification = AacSpecification.Mpeg4
                                                audioDescriptionBroadcasterMix =
AacAudioDescriptionBroadcasterMix.Normal
                                            }
                                    }
                                },
                            )
                    }

                // Create an OutputGroup
                val fileMp4 =
                    OutputGroup {
                        name = "File Group"
                        customName = "mp4"
                        outputGroupSettings =
                            OutputGroupSettings {
                                type = OutputGroupType.FileGroupSettings
                                fileGroupSettings =
                                    FileGroupSettings {
                                        destination = mp4Output
                                    }
                            }
                        outputs = listOf(theOutput)
                    }

                val containerSettings1 =
                    ContainerSettings {
                        container = ContainerType.Raw
                    }

                val thumbs =
                    OutputGroup {
                        name = "File Group"
                        customName = "thumbs"
                        outputGroupSettings =
                            OutputGroupSettings {
                                type = OutputGroupType.FileGroupSettings
                                fileGroupSettings =
```

```
                            FileGroupSettings {
                                destination = thumbsOutput
                            }
                    }

                outputs =
                    listOf(
                        Output {
                            extension = "jpg"

                            this.containerSettings = containerSettings1
                            videoDescription =
                                VideoDescription {
                                    scalingBehavior = ScalingBehavior.Default
                                    sharpness = 50
                                    antiAlias = AntiAlias.Enabled
                                    timecodeInsertion =
VideoTimecodeInsertion.Disabled
                                    colorMetadata = ColorMetadata.Insert
                                    dropFrameTimecode = DropFrameTimecode.Enabled
                                    codecSettings =
                                        VideoCodecSettings {
                                            codec = VideoCodec.FrameCapture
                                            frameCaptureSettings =
                                                FrameCaptureSettings {
                                                    framerateNumerator = 1
                                                    framerateDenominator = 1
                                                    maxCaptures = 10000000
                                                    quality = 80
                                                }
                                        }
                                }
                        },
                    )
            }

        val audioSelectors1: MutableMap<String, AudioSelector> = HashMap()
        audioSelectors1["Audio Selector 1"] =
            AudioSelector {
                defaultSelection = AudioDefaultSelection.Default
                offset = 0
            }

        val jobSettings =
```

```
            JobSettings {
                inputs =
                    listOf(
                        Input {
                            audioSelectors = audioSelectors1
                            videoSelector =
                                VideoSelector {
                                    colorSpace = ColorSpace.Follow
                                    rotate = InputRotate.Degree0
                                }
                            filterEnable = InputFilterEnable.Auto
                            filterStrength = 0
                            deblockFilter = InputDeblockFilter.Disabled
                            denoiseFilter = InputDenoiseFilter.Disabled
                            psiControl = InputPsiControl.UsePsi
                            timecodeSource = InputTimecodeSource.Embedded
                            fileInput = fileInputVal

                            outputGroups = listOf(appleHLS, thumbs, fileMp4)
                        },
                    )
            }

        val createJobRequest =
            CreateJobRequest {
                role = mcRoleARN
                settings = jobSettings
            }

        val createJobResponse = mediaConvert.createJob(createJobRequest)
        return createJobResponse.job?.id
    } catch (ex: MediaConvertException) {
        println(ex.message)
        mcClient.close()
        exitProcess(0)
    }
}

fun createOutput(
    nameModifierVal: String,
    segmentModifierVal: String,
    qvbrMaxBitrate: Int,
    qvbrQualityLevelVal: Int,
    originWidth: Int,
```

```
    originHeight: Int,
    targetWidth: Int,
): Output? {
    val targetHeight = (
        (originHeight * targetWidth / originWidth).toFloat().roundToInt() -
            (originHeight * targetWidth / originWidth).toFloat().roundToInt() % 4
        )

    var output: Output?
    try {
        val audio1 =
            AudioDescription {
                audioTypeControl = AudioTypeControl.FollowInput
                languageCodeControl = AudioLanguageCodeControl.FollowInput
                codecSettings =
                    AudioCodecSettings {
                        codec = AudioCodec.Aac
                        aacSettings =
                            AacSettings {
                                codecProfile = AacCodecProfile.Lc
                                rateControlMode = AacRateControlMode.Cbr
                                codingMode = AacCodingMode.CodingMode2_0
                                sampleRate = 44100
                                bitrate = 96000
                                rawFormat = AacRawFormat.None
                                specification = AacSpecification.Mpeg4
                                audioDescriptionBroadcasterMix =
 AacAudioDescriptionBroadcasterMix.Normal
                            }
                    }
            }

        output =
            Output {
                nameModifier = nameModifierVal
                outputSettings =
                    OutputSettings {
                        hlsSettings =
                            HlsSettings {
                                segmentModifier = segmentModifierVal
                                audioGroupId = "program_audio"
                                iFrameOnlyManifest = HlsIFrameOnlyManifest.Exclude
                            }
                    }
```

```
                containerSettings =
                    ContainerSettings {
                        container = ContainerType.M3U8
                        this.m3u8Settings =
                            M3u8Settings {
                                audioFramesPerPes = 4
                                pcrControl = M3u8PcrControl.PcrEveryPesPacket
                                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 = listOf(482, 483, 484, 485, 486, 487,
488, 489, 490, 491, 492)
                            }

                        videoDescription =
                            VideoDescription {
                                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 {
                                        codec = VideoCodec.H264
                                        h264Settings =
                                            H264Settings {
                                                rateControlMode =
H264RateControlMode.Qvbr
                                                parControl =
H264ParControl.InitializeFromSource
```

```
                                                qualityTuningLevel =
H264QualityTuningLevel.SinglePass
                                                qvbrSettings =
                                                    H264QvbrSettings {
                                                        qvbrQualityLevel =
qvbrQualityLevelVal
                                                    }
                                                codecLevel = H264CodecLevel.Auto
                                                codecProfile =
                                                    if (targetHeight > 720 &&
                                                        targetWidth > 1280
                                                    ) {
                                                        H264CodecProfile.High
                                                    } else {
                                                        H264CodecProfile.Main
                                                    }
                                                maxBitrate = qvbrMaxBitrate
                                                framerateControl =
H264FramerateControl.InitializeFromSource
                                                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.DuplicateDrop
                                                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
                                            }
                                        }
                                    audioDescriptions = listOf(audio1)
                                }
                            }
                        }
    } catch (ex: MediaConvertException) {
        println(ex.toString())
        exitProcess(0)
    }
    return output
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CreateJob」を参照してください。

`GetJob`

次の例は、`GetJob` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getSpecificJob(
    mcClient: MediaConvertClient,
    jobId: String?,
) {
    val describeEndpoints =
        DescribeEndpointsRequest {
            maxResults = 20
        }

    val res = mcClient.describeEndpoints(describeEndpoints)
    if (res.endpoints?.size!! <= 0) {
        println("Cannot find MediaConvert service endpoint URL!")
        exitProcess(0)
    }

    val endpointURL = res.endpoints!!.get(0).url!!
    val mediaConvert =
        MediaConvertClient.fromEnvironment {
            region = "us-west-2"
            endpointProvider =
                MediaConvertEndpointProvider {
                    Endpoint(endpointURL)
                }
        }

    val jobRequest =
        GetJobRequest {
            id = jobId
        }

    val response: GetJobResponse = mediaConvert.getJob(jobRequest)
    println("The ARN of the job is ${response.job?.arn}.")
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「GetJob」を参照してください。

ListJobs

次の例は、ListJobs を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listCompleteJobs(mcClient: MediaConvertClient) {
    val describeEndpoints =
        DescribeEndpointsRequest {
            maxResults = 20
        }

    val res = mcClient.describeEndpoints(describeEndpoints)
    if (res.endpoints?.size!! <= 0) {
        println("Cannot find MediaConvert service endpoint URL!")
        exitProcess(0)
    }
    val endpointURL = res.endpoints!![0].url!!
    val mediaConvert =
        MediaConvertClient.fromEnvironment {
            region = "us-west-2"
            endpointProvider =
                MediaConvertEndpointProvider {
                    Endpoint(endpointURL)
                }
        }

    val jobsRequest =
        ListJobsRequest {
            maxResults = 10
            status = JobStatus.fromValue("COMPLETE")
        }

    val jobsResponse = mediaConvert.listJobs(jobsRequest)
    val jobs = jobsResponse.jobs
    if (jobs != null) {
        for (job in jobs) {
            println("The JOB ARN is ${job.arn}")
        }
    }
```

```
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ListJobs」を参照してください。

SDK for Kotlin を使用した Amazon Pinpoint 例

次のコード例は、Amazon Pinpoint で AWS SDK for Kotlin 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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createApplication(applicationName: String?): String? {
    val createApplicationRequestOb =
        CreateApplicationRequest {
            name = applicationName
```

```
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.createApp(
                CreateAppRequest {
                    createApplicationRequest = createApplicationRequestOb
                },
            )
        return result.applicationResponse?.id
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateApp」を参照してください。

`CreateCampaign`

次の例は、`CreateCampaign` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createPinCampaign(
    appId: String,
    segmentIdVal: String,
) {
    val scheduleOb =
        Schedule {
            startTime = "IMMEDIATE"
        }

    val defaultMessageOb =
        Message {
            action = Action.OpenApp
```

```
            body = "My message body"
            title = "My message title"
        }

    val messageConfigurationOb =
        MessageConfiguration {
            defaultMessage = defaultMessageOb
        }

    val writeCampaign =
        WriteCampaignRequest {
            description = "My description"
            schedule = scheduleOb
            name = "MyCampaign"
            segmentId = segmentIdVal
            messageConfiguration = messageConfigurationOb
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result: CreateCampaignResponse =
            pinpoint.createCampaign(
                CreateCampaignRequest {
                    applicationId = appId
                    writeCampaignRequest = writeCampaign
                },
            )
        println("Campaign ID is ${result.campaignResponse?.id}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CreateCampaign」を参照してください。

`CreateSegment`

次の例は、`CreateSegment` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createPinpointSegment(applicationIdVal: String?): String? {
    val segmentAttributes = mutableMapOf<String, AttributeDimension>()
    val myList = mutableListOf<String>()
    myList.add("Lakers")

    val atts =
        AttributeDimension {
            attributeType = AttributeType.Inclusive
            values = myList
        }

    segmentAttributes["Team"] = atts
    val recencyDimension =
        RecencyDimension {
            duration = Duration.fromValue("DAY_30")
            recencyType = RecencyType.fromValue("ACTIVE")
        }

    val segmentBehaviors =
        SegmentBehaviors {
            recency = recencyDimension
        }

    val segmentLocation = SegmentLocation {}
    val dimensionsOb =
        SegmentDimensions {
            attributes = segmentAttributes
            behavior = segmentBehaviors
            demographic = SegmentDemographics {}
            location = segmentLocation
        }

    val writeSegmentRequestOb =
        WriteSegmentRequest {
```

```
            name = "MySegment101"
            dimensions = dimensionsOb
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val createSegmentResult: CreateSegmentResponse =
            pinpoint.createSegment(
                CreateSegmentRequest {
                    applicationId = applicationIdVal
                    writeSegmentRequest = writeSegmentRequestOb
                },
            )
        println("Segment ID is ${createSegmentResult.segmentResponse?.id}")
        return createSegmentResult.segmentResponse?.id
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CreateSegment」を参照してください。

`DeleteApp`

次の例は、`DeleteApp` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deletePinApp(appId: String?) {
    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.deleteApp(
                DeleteAppRequest {
                    applicationId = appId
                },
            )
```

```
        val appName = result.applicationResponse?.name
        println("Application $appName has been deleted.")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DeleteApp」を参照してください。

`DeleteEndpoint`

次の例は、`DeleteEndpoint` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deletePinEncpoint(
    appIdVal: String?,
    endpointIdVal: String?,
) {
    val deleteEndpointRequest =
        DeleteEndpointRequest {
            applicationId = appIdVal
            endpointId = endpointIdVal
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.deleteEndpoint(deleteEndpointRequest)
        val id = result.endpointResponse?.id
        println("The deleted endpoint is  $id")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DeleteEndpoint」を参照してください。

GetEndpoint

次の例は、GetEndpoint を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun lookupPinpointEndpoint(
    appId: String?,
    endpoint: String?,
) {
    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.getEndpoint(
                GetEndpointRequest {
                    applicationId = appId
                    endpointId = endpoint
                },
            )
        val endResponse = result.endpointResponse

        // Uses the Google Gson library to pretty print the endpoint JSON.
        val gson: com.google.gson.Gson =
            GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting()
                .create()

        val endpointJson: String = gson.toJson(endResponse)
        println(endpointJson)
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「GetEndpoint」を参照してください。

GetSegments

次の例は、GetSegments を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listSegs(appId: String?) {
    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val response =
            pinpoint.getSegments(
                GetSegmentsRequest {
                    applicationId = appId
                },
            )
        response.segmentsResponse?.item?.forEach { segment ->
            println("Segement id is ${segment.id}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「GetSegments」を参照してください。

SendMessages

次の例は、SendMessages を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

val body: String =
    """
    Amazon Pinpoint test (AWS SDK for Kotlin)

    This email was sent through the Amazon Pinpoint Email API using the AWS SDK for
 Kotlin.

    """.trimIndent()

suspend fun main(args: Array<String>) {
    val 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
        toAddress - The to address. This address has to be verified in Amazon
 Pinpoint in the region you're using to send email
    """

    if (args.size != 3) {
        println(usage)
        exitProcess(0)
    }

    val subject = args[0]
    val senderAddress = args[1]
    val toAddress = args[2]
    sendEmail(subject, senderAddress, toAddress)
}

suspend fun sendEmail(
```

```
    subjectVal: String?,
    senderAddress: String,
    toAddressVal: String,
) {
    var content =
        Content {
            data = body
        }

    val messageBody =
        Body {
            text = content
        }

    val subContent =
        Content {
            data = subjectVal
        }

    val message =
        Message {
            body = messageBody
            subject = subContent
        }

    val destinationOb =
        Destination {
            toAddresses = listOf(toAddressVal)
        }

    val emailContent =
        EmailContent {
            simple = message
        }

    val sendEmailRequest =
        SendEmailRequest {
            fromEmailAddress = senderAddress
            destination = destinationOb
            this.content = emailContent
        }

    PinpointEmailClient { region = "us-east-1" }.use { pinpointemail ->
        pinpointemail.sendEmail(sendEmailRequest)
```

```
        println("Message Sent")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「SendMessages」を参照してください。

SDK for Kotlin を使用した Amazon RDS の例

次のコード例は、Amazon RDS で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、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、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

トピック

- 基本
- アクション
- シナリオ

基本

基本を学ぶ

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
Before running this code example, set up your development environment, including
 your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.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

This example performs the following tasks:

1. Returns a list of the available DB engines by invoking the
 DescribeDbEngineVersions method.
2. Selects an engine family and create a custom DB parameter group by invoking the
 createDBParameterGroup method.
3. Gets the parameter groups by invoking the DescribeDbParameterGroups method.
4. Gets parameters in the group by invoking the DescribeDbParameters method.
5. Modifies both the auto_increment_offset and auto_increment_increment parameters
 by invoking the modifyDbParameterGroup method.
6. Gets and displays the updated parameters.
7. Gets a list of allowed engine versions by invoking the describeDbEngineVersions
 method.
8. Gets a list of micro instance classes available for the selected engine.
```

```
9. Creates an Amazon Relational Database Service (Amazon RDS) database instance that
 contains a MySQL database and uses the parameter group.
10. Waits for DB instance to be ready and prints out the connection endpoint value.
11. Creates a snapshot of the DB instance.
12. Waits for the DB snapshot to be ready.
13. Deletes the DB instance.
14. Deletes the parameter group.
 */

var sleepTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier> <dbName>
 <dbSnapshotIdentifier><secretName>

        Where:
            dbGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
 the database credentials.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val dbGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceIdentifier = args[2]
    val dbName = args[3]
    val dbSnapshotIdentifier = args[4]
    val secretName = args[5]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
 User::class.java)
    val username = user.username
    val userPassword = user.password
```

```
    println("1. Return a list of the available DB engines")
    describeDBEngines()

    println("2. Create a custom parameter group")
    createDBParameterGroup(dbGroupName, dbParameterGroupFamily)

    println("3. Get the parameter groups")
    describeDbParameterGroups(dbGroupName)

    println("4. Get the parameters in the group")
    describeDbParameters(dbGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBParas(dbGroupName)

    println("6. Display the updated value")
    describeDbParameters(dbGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedEngines(dbParameterGroupFamily)

    println("8. Get a list of micro instance classes available for the selected
engine")
    getMicroInstances()

    println("9. Create an RDS database instance that contains a MySql database and
uses the parameter group")
    val dbARN = createDatabaseInstance(dbGroupName, dbInstanceIdentifier, dbName,
username, userPassword)
    println("The ARN of the new database is $dbARN")

    println("10. Wait for DB instance to be ready")
    waitForDbInstanceReady(dbInstanceIdentifier)

    println("11. Create a snapshot of the DB instance")
    createDbSnapshot(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("12. Wait for DB snapshot to be ready")
    waitForSnapshotReady(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("13. Delete the DB instance")
    deleteDbInstance(dbInstanceIdentifier)
```

```
    println("14. Delete the parameter group")
    if (dbARN != null) {
        deleteParaGroup(dbGroupName, dbARN)
    }

    println("The Scenario has successfully completed.")
}

suspend fun deleteParaGroup(
    dbGroupName: String,
    dbARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false // Reset this value.
            didFind = false // Reset this value.
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(dbARN) == 0) {
                        println("$dbARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database name.
                        isDataDel = true
                    }
                    index++
                }
            }
        }

        // Delete the para group.
        val parameterGroupRequest =
```

```
            DeleteDbParameterGroupRequest {
                dbParameterGroupName = dbGroupName
            }
        rdsClient.deleteDbParameterGroup(parameterGroupRequest)
        println("$dbGroupName was deleted.")
    }
}

suspend fun deleteDbInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}

// Waits until the snapshot instance is available.
suspend fun waitForSnapshotReady(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbSnapshotsRequest {
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    while (!snapshotReady) {
        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbSnapshots(snapshotsRequest)
            val snapshotList: List<DbSnapshot>? = response.dbSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
```

```
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

// Create an Amazon RDS snapshot.
suspend fun createDbSnapshot(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    val snapshotRequest =
        CreateDbSnapshotRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbSnapshot(snapshotRequest)
        print("The Snapshot id is ${response.dbSnapshot?.dbiResourceId}")
    }
}

// Waits until the database instance is available.
suspend fun waitForDbInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }
    var endpoint = ""
    while (!instanceReady) {
        RdsClient { region = "us-west-2" }.use { rdsClient ->
```

```
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint?.address.toString()
                        instanceReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}

// Create a database instance and return the ARN of the database.
suspend fun createDatabaseInstance(
    dbGroupNameVal: String?,
    dbInstanceIdentifierVal: String?,
    dbNameVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNameVal
            dbParameterGroupName = dbGroupNameVal
            engine = "mysql"
            dbInstanceClass = "db.t3.micro"
            engineVersion = "8.0.35"
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
```

```
        return response.dbInstance?.dbInstanceArn
    }
}

// Get a list of micro instances.
suspend fun getMicroInstances() {
    val dbInstanceOptionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "mysql"
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeOrderableDbInstanceOptions(dbInstanceOptionsRequest)
        val orderableDBInstances = response.orderableDbInstanceOptions
        if (orderableDBInstances != null) {
            for (dbInstanceOption in orderableDBInstances) {
                println("The engine version is ${dbInstanceOption.engineVersion}")
                println("The engine description is ${dbInstanceOption.engine}")
            }
        }
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "mysql"
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        val dbEngines: List<DbEngineVersion>? = response.dbEngineVersions
        if (dbEngines != null) {
            for (dbEngine in dbEngines) {
                println("The engine version is ${dbEngine.engineVersion}")
                println("The engine description is ${dbEngine.dbEngineDescription}")
            }
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBParas(dbGroupName: String) {
```

```
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.Immediate
            parameterValue = "5"
        }

    val paraList: ArrayList<Parameter> = ArrayList()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            parameters = paraList
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbParameterGroup(groupRequest)
        println("The parameter group ${response.dbParameterGroupName} was
 successfully modified")
    }
}

// Retrieve parameters in the group.
suspend fun describeDbParameters(
    dbGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
            }
        } else {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
                source = "user"
            }
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameters(dbParameterGroupsRequest)
        val dbParameters: List<Parameter>? = response.parameters
        var paraName: String
        if (dbParameters != null) {
```

```
            for (para in dbParameters) {
                // Only print out information about either auto_increment_offset or
 auto_increment_increment.
                paraName = para.parameterName.toString()
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    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}")
                }
            }
        }
    }
}

suspend fun describeDbParameterGroups(dbGroupName: String?) {
    val groupsRequest =
        DescribeDbParameterGroupsRequest {
            dbParameterGroupName = dbGroupName
            maxRecords = 20
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameterGroups(groupsRequest)
        val groups = response.dbParameterGroups
        if (groups != null) {
            for (group in groups) {
                println("The group name is ${group.dbParameterGroupName}")
                println("The group description is ${group.description}")
            }
        }
    }
}

// Create a parameter group.
suspend fun createDBParameterGroup(
    dbGroupName: String?,
    dbParameterGroupFamilyVal: String?,
) {
```

```
    val groupRequest =
        CreateDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbParameterGroup(groupRequest)
        println("The group name is
 ${response.dbParameterGroup?.dbParameterGroupName}")
    }
}

// Returns a list of the available DB engines.
suspend fun describeDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            defaultOnly = true
            engine = "mysql"
            maxRecords = 20
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        val engines: List<DbEngineVersion>? = response.dbEngineVersions

        // Get all DbEngineVersion objects.
        if (engines != null) {
            for (engineOb in engines) {
                println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}.")
                println("The name of the database engine ${engineOb.engine}.")
                println("The version number of the database engine
 ${engineOb.engineVersion}")
            }
        }
    }
}

suspend fun getSecretValues(secretName: String?): String? {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
```

```
        }

    SecretsManagerClient { region = "us-west-2" }.use { secretsClient ->
        val valueResponse = secretsClient.getSecretValue(valueRequest)
        return valueResponse.secretString
    }
}
```

- API の詳細については、「AWS SDK for Kotlin 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

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createDatabaseInstance(
    dbInstanceIdentifierVal: String?,
    dbNamedbVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNamedbVal
            engine = "mysql"
            dbInstanceClass = "db.t3.micro" // Use a supported instance class
            engineVersion = "8.0.39" // Use a supported engine version
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
    }
}

// Waits until the database instance is available.
suspend fun waitForInstanceReady(dbInstanceIdentifierVal: String?) {
    val sleepTime: Long = 20
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
```

```
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true
                    } else {
                        println("...$instanceReadyStr")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
        println("Database instance is available!")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CreateDBInstance」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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteDatabaseInstance(dbInstanceIdentifierVal: String?) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
```

```
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DeleteDBInstance」を参照してください。

`DescribeAccountAttributes`

次の例は、`DescribeAccountAttributes` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getAccountAttributes() {
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeAccountAttributes(DescribeAccountAttributesRequest {})
        response.accountQuotas?.forEach { quotas ->
            val response = response.accountQuotas
            println("Name is: ${quotas.accountQuotaName}")
            println("Max value is ${quotas.max}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeAccountAttributes」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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeInstances() {
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbInstances(DescribeDbInstancesRequest {})
        response.dbInstances?.forEach { instance ->
            println("Instance Identifier is ${instance.dbInstanceIdentifier}")
            println("The Engine is ${instance.engine}")
            println("Connection endpoint is ${instance.endpoint?.address}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeDBInstances」を参照してください。

`ModifyDBInstance`

次の例は、`ModifyDBInstance` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun updateIntance(
    dbInstanceIdentifierVal: String?,
    masterUserPasswordVal: String?,
) {
```

```
    val request =
        ModifyDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            publiclyAccessible = true
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val instanceResponse = rdsClient.modifyDbInstance(request)
        println("The ARN of the modified database is
 ${instanceResponse.dbInstance?.dbInstanceArn}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ModifyDBInstance」を参照してください。

シナリオ

Aurora Serverless 作業項目トラッカーの作成

次のコード例は、Amazon Aurora Serverless データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信する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。

SDK for Kotlin

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 サーバーレスデータをクエリする Spring REST API と React アプリケーションで使用するための完全なソースコードと設定方法については、GitHub にある完全な例を参照してください。

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

SDK for Kotlin を使用した Amazon RDS Data Service の例

次のコード例は、Amazon RDS Data Service で AWS SDK for Kotlin 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。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- シナリオ

シナリオ

Aurora Serverless 作業項目トラッカーの作成

次のコード例は、Amazon Aurora Serverless データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信する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。

SDK for Kotlin

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 サーバーレスデータをクエリする Spring REST API と React アプリケーションで使用するための完全なソースコードと設定方法については、GitHub にある完全な例を参照してください。

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

SDK for Kotlin を使用した Amazon Redshift の例

次のコード例は、Amazon Redshift で AWS SDK for Kotlin 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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する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

`CreateCluster`

次の例は、`CreateCluster` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

クラスターを作成します。

```
suspend fun createCluster(
```

```
    clusterId: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val clusterRequest =
        CreateClusterRequest {
            clusterIdentifier = clusterId
            availabilityZone = "us-east-1a"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
            nodeType = "ra3.4xlarge"
            publiclyAccessible = true
            numberOfNodes = 2
        }

    RedshiftClient { region = "us-east-1" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.createCluster(clusterRequest)
        println("Created cluster ${clusterResponse.cluster?.clusterIdentifier}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CreateCluster」を参照してください。

`DeleteCluster`

次の例は、`DeleteCluster` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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。

```
suspend fun deleteRedshiftCluster(clusterId: String?) {
    val request =
```

```
        DeleteClusterRequest {
            clusterIdentifier = clusterId
            skipFinalClusterSnapshot = true
        }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val response = redshiftClient.deleteCluster(request)
        println("The status is ${response.cluster?.clusterStatus}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DeleteCluster」を参照してください。

`DescribeClusters`

次の例は、`DescribeClusters` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

クラスターを記述します。

```
suspend fun describeRedshiftClusters() {
    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse =
 redshiftClient.describeClusters(DescribeClustersRequest {})
        val clusterList = clusterResponse.clusters

        if (clusterList != null) {
            for (cluster in clusterList) {
                println("Cluster database name is ${cluster.dbName}")
                println("Cluster status is ${cluster.clusterStatus}")
            }
        }
```

```
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeClusters」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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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。

```
suspend fun modifyCluster(clusterId: String?) {
    val modifyClusterRequest =
        ModifyClusterRequest {
            clusterIdentifier = clusterId
            preferredMaintenanceWindow = "wed:07:30-wed:08:00"
        }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.modifyCluster(modifyClusterRequest)
        println(
            "The modified cluster was successfully modified and has
 ${clusterResponse.cluster?.preferredMaintenanceWindow} as the maintenance window",
        )
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ModifyCluster」を参照してください。

シナリオ

Amazon Redshift データ追跡用のウェブアプリケーションの作成

次のコード例は、Amazon Redshift データベースを使用して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示しています。

SDK for Kotlin

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 サーバーレスデータをクエリする Spring REST API と React アプリケーションで使用するための完全なソースコードと設定方法については、GitHub にある詳しい事例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 Redshift
- Amazon SES

SDK for Kotlin を使用する Amazon Rekognition の例

次のコード例は、Amazon Rekognition で AWS SDK for Kotlin 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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun compareTwoFaces(
    similarityThresholdVal: Float,
    sourceImageVal: String,
    targetImageVal: String,
) {
    val sourceBytes = (File(sourceImageVal).readBytes())
    val targetBytes = (File(targetImageVal).readBytes())

    // Create an Image object for the source image.
    val souImage =
        Image {
            bytes = sourceBytes
        }

    val tarImage =
        Image {
            bytes = targetBytes
        }

    val facesRequest =
        CompareFacesRequest {
            sourceImage = souImage
            targetImage = tarImage
            similarityThreshold = similarityThresholdVal
        }
```

```
    RekognitionClient { region = "us-east-1" }.use { rekClient ->

        val compareFacesResult = rekClient.compareFaces(facesRequest)
        val faceDetails = compareFacesResult.faceMatches

        if (faceDetails != null) {
            for (match: CompareFacesMatch in faceDetails) {
                val face = match.face
                val position = face?.boundingBox
                if (position != null) {
                    println("Face at ${position.left} ${position.top} matches with
 ${face.confidence} % confidence.")
                }
            }
        }

        val uncompared = compareFacesResult.unmatchedFaces
        if (uncompared != null) {
            println("There was ${uncompared.size} face(s) that did not match")
        }

        println("Source image rotation:
 ${compareFacesResult.sourceImageOrientationCorrection}")
        println("target image rotation:
 ${compareFacesResult.targetImageOrientationCorrection}")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API Reference」の「CompareFaces」を参照してください。

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createMyCollection(collectionIdVal: String) {
    val request =
        CreateCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API Reference」の「CreateCollection」を参照してください。

`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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteMyCollection(collectionIdVal: String) {
```

```
    val request =
        DeleteCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API Reference」の「DeleteCollection」を参照してください。

`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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteFacesCollection(
    collectionIdVal: String?,
    faceIdVal: String,
) {
    val deleteFacesRequest =
        DeleteFacesRequest {
            collectionId = collectionIdVal
            faceIds = listOf(faceIdVal)
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
        println("$faceIdVal was deleted from the collection")
```

```
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API Reference」の「DeleteFaces」を参照してください。

`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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeColl(collectionName: String) {
    val request =
        DescribeCollectionRequest {
            collectionId = collectionName
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.describeCollection(request)
        println("The collection Arn is ${response.collectionArn}")
        println("The collection contains this many faces ${response.faceCount}")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API Reference」の「DescribeCollection」を参照してください。

`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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun detectFacesinImage(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectFacesRequest {
            attributes = listOf(Attribute.All)
            image = souImage
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectFaces(request)
        response.faceDetails?.forEach { face ->
            val ageRange = face.ageRange
            println("The detected face is estimated to be between ${ageRange?.low}
 and ${ageRange?.high} years old.")
            println("There is a smile ${face.smile?.value}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API Reference」の「DetectFaces」を参照してください。

`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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun detectImageLabels(sourceImage: String) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }
    val request =
        DetectLabelsRequest {
            image = souImage
            maxLabels = 10
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectLabels(request)
        response.labels?.forEach { label ->
            println("${label.name} : ${label.confidence}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API Reference」の「DetectLabels」を参照してください。

`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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun detectModLabels(sourceImage: String) {
    val myImage =
        Image {
            this.bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectModerationLabelsRequest {
            image = myImage
            minConfidence = 60f
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectModerationLabels(request)
        response.moderationLabels?.forEach { label ->
            println("Label: ${label.name} - Confidence: ${label.confidence} %
 Parent: ${label.parentName}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API Reference」の「DetectModerationLabels」を参照してください。

`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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun detectTextLabels(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectTextRequest {
            image = souImage
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectText(request)
        response.textDetections?.forEach { text ->
            println("Detected: ${text.detectedText}")
            println("Confidence: ${text.confidence}")
            println("Id: ${text.id}")
            println("Parent Id:  ${text.parentId}")
            println("Type: ${text.type}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API Reference」の「DetectText」を参照してください。

`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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun addToCollection(
    collectionIdVal: String?,
    sourceImage: String,
) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        IndexFacesRequest {
            collectionId = collectionIdVal
            image = souImage
            maxFaces = 1
            qualityFilter = QualityFilter.Auto
            detectionAttributes = listOf(Attribute.Default)
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val facesResponse = rekClient.indexFaces(request)

        // Display the results.
        println("Results for the image")
        println("\n Faces indexed:")
        facesResponse.faceRecords?.forEach { faceRecord ->
            println("Face ID: ${faceRecord.face?.faceId}")
            println("Location: ${faceRecord.faceDetail?.boundingBox}")
        }

        println("Faces not indexed:")
        facesResponse.unindexedFaces?.forEach { unindexedFace ->
            println("Location: ${unindexedFace.faceDetail?.boundingBox}")
            println("Reasons:")
```

```
            unindexedFace.reasons?.forEach { reason ->
                println("Reason:  $reason")
            }
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API Reference」の「IndexFaces」を参照してください。

`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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listAllCollections() {
    val request =
        ListCollectionsRequest {
            maxResults = 10
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API Reference」の「ListCollections」を参照してください。

`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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listFacesCollection(collectionIdVal: String?) {
    val request =
        ListFacesRequest {
            collectionId = collectionIdVal
            maxResults = 10
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listFaces(request)
        response.faces?.forEach { face ->
            println("Confidence level there is a face: ${face.confidence}")
            println("The face Id value is ${face.faceId}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API Reference」の「ListFaces」を参照してください。

`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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        RecognizeCelebritiesRequest {
            image = souImage
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.recognizeCelebrities(request)
        response.celebrityFaces?.forEach { celebrity ->
            println("Celebrity recognized: ${celebrity.name}")
            println("Celebrity ID:${celebrity.id}")
            println("Further information (if available):")
            celebrity.urls?.forEach { url ->
                println(url)
            }
        }
        println("${response.unrecognizedFaces?.size} face(s) were unrecognized.")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「RecognizeCelebrities」を参照してください。

シナリオ

サーバーレスアプリケーションを作成して写真の管理

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

SDK for Kotlin

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設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、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

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

Amazon S3 バケットに保存されたビデオ内の顔を検出する

```
suspend fun startFaceDetection(
    channelVal: NotificationChannel?,
    bucketVal: String,
    videoVal: String,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }

    val request =
        StartFaceDetectionRequest {
            jobTag = "Faces"
            faceAttributes = FaceAttributes.All
            notificationChannel = channelVal
            video = vidOb
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val startLabelDetectionResult = rekClient.startFaceDetection(request)
        startJobId = startLabelDetectionResult.jobId.toString()
    }
}

suspend fun getFaceResults() {
    var finished = false
    var status: String
    var yy = 0
```

```
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var response: GetFaceDetectionResponse? = null

        val recognitionRequest =
            GetFaceDetectionRequest {
                jobId = startJobId
                maxResults = 10
            }

        // Wait until the job succeeds.
        while (!finished) {
            response = rekClient.getFaceDetection(recognitionRequest)
            status = response.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = response?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        // Show face information.
        response?.faces?.forEach { face ->
            println("Age: ${face.face?.ageRange}")
            println("Face: ${face.face?.beard}")
            println("Eye glasses: ${face?.face?.eyeglasses}")
            println("Mustache: ${face.face?.mustache}")
            println("Smile: ${face.face?.smile}")
        }
    }
}
```

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。

```
suspend fun startModerationDetection(
    channel: NotificationChannel?,
    bucketVal: String?,
    videoVal: String?,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }
    val request =
        StartContentModerationRequest {
            jobTag = "Moderation"
            notificationChannel = channel
            video = vidOb
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val startModDetectionResult = rekClient.startContentModeration(request)
        startJobId = startModDetectionResult.jobId.toString()
    }
}

suspend fun getModResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var modDetectionResponse: GetContentModerationResponse? = null

        val modRequest =
            GetContentModerationRequest {
                jobId = startJobId
                maxResults = 10
            }

        // Wait until the job succeeds.
        while (!finished) {
            modDetectionResponse = rekClient.getContentModeration(modRequest)
```

```
            status = modDetectionResponse.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = modDetectionResponse?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        modDetectionResponse?.moderationLabels?.forEach { mod ->
            val seconds: Long = mod.timestamp / 1000
            print("Mod label: $seconds ")
            println(mod.moderationLabel)
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin 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

イメージ内のオブジェクトを検出する

次のコード例は、Amazon Rekognition を使用してイメージ内のカテゴリ別にオブジェクトを検出するアプリケーションを構築する方法を示しています。

SDK for Kotlin

Amazon Simple Storage Service (Amazon S3) バケットにある画像内からカテゴリ別にオブジェクトを Amazon Rekognition を使用して識別するアプリケーションを、Amazon Rekognition Kotlin API を使用して作成する方法を示します。アプリケーションは、Amazon Simple Email Service (Amazon SES) を使用して、結果を記載した 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および実行の手順については、GitHub で完全な例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 Rekognition
- Amazon S3
- Amazon SES

SDK for Kotlin を使用した Route 53 ドメイン登録の例

次のコード例は、Route 53 ドメイン登録で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
           <domainType>

       Where:
           domainType - The domain type (for example, com).
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val domainType = args[0]
    println("Invokes ListPrices using a Paginated method.")
    listPricesPaginated(domainType)
}

suspend fun listPricesPaginated(domainType: String) {
    val pricesRequest =
        ListPricesRequest {
            maxItems = 10
```

```
            tld = domainType
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}
```

- API の詳細については、「AWS API リファレンス」の「ListPrices」を参照してください。

トピック

- 基本
- アクション

基本

基本を学ぶ

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin 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.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <domainType> <phoneNumber> <email> <domainSuggestion> <firstName>
 <lastName> <city>
        Where:
           domainType - The domain type (for example, com).
           phoneNumber - The phone number to use (for example, +1.2065550100)
```

```
        email - The email address to use.
        domainSuggestion - The domain suggestion (for example, findmy.example).
        firstName - The first name to use to register a domain.
        lastName -  The last name to use to register a domain.
        city - The city to use to register a domain.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val domainType = args[0]
    val phoneNumber = args[1]
    val email = args[2]
    val domainSuggestion = args[3]
    val firstName = args[4]
    val lastName = args[5]
    val city = args[6]

    println(DASHES)
    println("Welcome to the Amazon Route 53 domains example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List current domains.")
    listDomains()
    println(DASHES)

    println(DASHES)
    println("2. List operations in the past year.")
    listOperations()
    println(DASHES)

    println(DASHES)
    println("3. View billing for the account in the past year.")
    listBillingRecords()
    println(DASHES)

    println(DASHES)
    println("4. View prices for domain types.")
    listAllPrices(domainType)
    println(DASHES)
```

```
    println(DASHES)
    println("5. Get domain suggestions.")
    listDomainSuggestions(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("6. Check domain availability.")
    checkDomainAvailability(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("7. Check domain transferability.")
    checkDomainTransferability(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("8. Request a domain registration.")
    val opId = requestDomainRegistration(domainSuggestion, phoneNumber, email,
 firstName, lastName, city)
    println(DASHES)

    println(DASHES)
    println("9. Get operation details.")
    getOperationalDetail(opId)
    println(DASHES)

    println(DASHES)
    println("10. Get domain details.")
    println("Note: You must have a registered domain to get details.")
    println("Otherwise an exception is thrown that states ")
    println("Domain xxxxxxx not found in xxxxxxx account.")
    getDomainDetails(domainSuggestion)
    println(DASHES)
}

suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getDomainDetail(detailRequest)
        println("The contact first name is
 ${response.registrantContact?.firstName}")
```

```
        println("The contact last name is ${response.registrantContact?.lastName}")
        println("The contact org name is
 ${response.registrantContact?.organizationName}")
    }
}

suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}

suspend fun requestDomainRegistration(
    domainSuggestion: String?,
    phoneNumberVal: String?,
    emailVal: String?,
    firstNameVal: String?,
    lastNameVal: String?,
    cityVal: String?,
): String? {
    val contactDetail =
        ContactDetail {
            contactType = ContactType.Company
            state = "LA"
            countryCode = CountryCode.In
            email = emailVal
            firstName = firstNameVal
            lastName = lastNameVal
            city = cityVal
            phoneNumber = phoneNumberVal
            organizationName = "My Org"
            addressLine1 = "My Address"
            zipCode = "123 123"
        }

    val domainRequest =
        RegisterDomainRequest {
            adminContact = contactDetail
            registrantContact = contactDetail
```

```
            techContact = contactDetail
            domainName = domainSuggestion
            autoRenew = true
            durationInYears = 1
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}

suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}

suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
}

suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
```

```
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getDomainSuggestions(suggestionsRequest)
        response.suggestionsList?.forEach { suggestion ->
            println("Suggestion Name: ${suggestion.domainName}")
            println("Availability: ${suggestion.availability}")
            println(" ")
        }
    }
}

suspend fun listAllPrices(domainType: String?) {
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}

suspend fun listBillingRecords() {
    val currentDate = Date()
    val localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    val localDateTime2 = localDateTime.minusYears(1)
    val myStartTime = localDateTime2.toInstant(zoneOffset)
    val myEndTime = localDateTime.toInstant(zoneOffset)
    val timeStart: Instant? = myStartTime?.let { Instant(it) }
    val timeEnd: Instant? = myEndTime?.let { Instant(it) }
```

```
    val viewBillingRequest =
        ViewBillingRequest {
            start = timeStart
            end = timeEnd
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .viewBillingPaginated(viewBillingRequest)
            .transform { it.billingRecords?.forEach { obj -> emit(obj) } }
            .collect { billing ->
                println("Bill Date: ${billing.billDate}")
                println("Operation: ${billing.operation}")
                println("Price: ${billing.price}")
            }
    }
}

suspend fun listOperations() {
    val currentDate = Date()
    var localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    localDateTime = localDateTime.minusYears(1)
    val myTime: java.time.Instant? = localDateTime.toInstant(zoneOffset)
    val time2: Instant? = myTime?.let { Instant(it) }
    val operationsRequest =
        ListOperationsRequest {
            submittedSince = time2
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listOperationsPaginated(operationsRequest)
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("Operation Id: ${content.operationId}")
                println("Status: ${content.status}")
                println("Date: ${content.submittedDate}")
            }
    }
}

suspend fun listDomains() {
```

```
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listDomainsPaginated(ListDomainsRequest {})
            .transform { it.domains?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("The domain name is ${content.domainName}")
            }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin 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

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CheckDomainAvailability」を参照してください。

`CheckDomainTransferability`

次の例は、`CheckDomainTransferability` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CheckDomainTransferability」を参照してください。

GetDomainDetail

次の例は、GetDomainDetail を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getDomainDetail(detailRequest)
        println("The contact first name is
 ${response.registrantContact?.firstName}")
        println("The contact last name is ${response.registrantContact?.lastName}")
        println("The contact org name is
 ${response.registrantContact?.organizationName}")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetDomainDetail」を参照してください。

GetDomainSuggestions

次の例は、GetDomainSuggestions を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getDomainSuggestions(suggestionsRequest)
        response.suggestionsList?.forEach { suggestion ->
            println("Suggestion Name: ${suggestion.domainName}")
            println("Availability: ${suggestion.availability}")
            println(" ")
        }
    }
}
```

- API の詳細については、「AWS API リファレンス」の「GetDomainSuggestions」を参照してください。

`GetOperationDetail`

次の例は、`GetOperationDetail` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}
```

- API の詳細については、AWS API リファレンスの「GetOperationDetail」を参照してください。

`ListDomains`

次の例は、`ListDomains` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listDomains() {
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listDomainsPaginated(ListDomainsRequest {})
            .transform { it.domains?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("The domain name is ${content.domainName}")
            }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ListDomains」を参照してください。

`ListOperations`

次の例は、`ListOperations` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listOperations() {
    val currentDate = Date()
    var localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    localDateTime = localDateTime.minusYears(1)
    val myTime: java.time.Instant? = localDateTime.toInstant(zoneOffset)
    val time2: Instant? = myTime?.let { Instant(it) }
    val operationsRequest =
        ListOperationsRequest {
            submittedSince = time2
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listOperationsPaginated(operationsRequest)
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("Operation Id: ${content.operationId}")
                println("Status: ${content.status}")
                println("Date: ${content.submittedDate}")
            }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ListOperations」を参照してください。

`ListPrices`

次の例は、`ListPrices` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listAllPrices(domainType: String?) {
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}
```

- API の詳細については、「AWS API リファレンス」の「ListPrices」を参照してください。

RegisterDomain

次の例は、RegisterDomain を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun requestDomainRegistration(
    domainSuggestion: String?,
    phoneNumberVal: String?,
    emailVal: String?,
    firstNameVal: String?,
    lastNameVal: String?,
    cityVal: String?,
): String? {
    val contactDetail =
        ContactDetail {
            contactType = ContactType.Company
            state = "LA"
            countryCode = CountryCode.In
            email = emailVal
            firstName = firstNameVal
            lastName = lastNameVal
            city = cityVal
            phoneNumber = phoneNumberVal
            organizationName = "My Org"
            addressLine1 = "My Address"
            zipCode = "123 123"
        }

    val domainRequest =
        RegisterDomainRequest {
            adminContact = contactDetail
            registrantContact = contactDetail
            techContact = contactDetail
            domainName = domainSuggestion
            autoRenew = true
            durationInYears = 1
```

```
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「RegisterDomain」を参照してください。

`ViewBilling`

次の例は、`ViewBilling` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listBillingRecords() {
    val currentDate = Date()
    val localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    val localDateTime2 = localDateTime.minusYears(1)
    val myStartTime = localDateTime2.toInstant(zoneOffset)
    val myEndTime = localDateTime.toInstant(zoneOffset)
    val timeStart: Instant? = myStartTime?.let { Instant(it) }
    val timeEnd: Instant? = myEndTime?.let { Instant(it) }

    val viewBillingRequest =
        ViewBillingRequest {
            start = timeStart
            end = timeEnd
        }
```

```
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .viewBillingPaginated(viewBillingRequest)
            .transform { it.billingRecords?.forEach { obj -> emit(obj) } }
            .collect { billing ->
                println("Bill Date: ${billing.billDate}")
                println("Operation: ${billing.operation}")
                println("Price: ${billing.price}")
            }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ViewBilling」を参照してください。

SDK for Kotlin を使用した Amazon S3 の例

次のコード例は、Amazon S3 で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、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、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

トピック

- 基本
- アクション
- シナリオ

基本

基本を学ぶ

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun main(args: Array<String>) {
    val 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).
        """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }
```

```
    val bucketName = args[0]
    val key = args[1]
    val objectPath = args[2]
    val savePath = args[3]
    val toBucket = args[4]

    // Create an Amazon S3 bucket.
    createBucket(bucketName)

    // Update a local file to the Amazon S3 bucket.
    putObject(bucketName, key, objectPath)

    // Download the object to another local file.
    getObjectFromMrap(bucketName, key, savePath)

    // List all objects located in the Amazon S3 bucket.
    listBucketObs(bucketName)

    // Copy the object to another Amazon S3 bucket
    copyBucketOb(bucketName, key, toBucket)

    // Delete the object from the Amazon S3 bucket.
    deleteBucketObs(bucketName, key)

    // Delete the Amazon S3 bucket.
    deleteBucket(bucketName)
    println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
```

```
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            this.body = Paths.get(objectPath).asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObjectFromMrap(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucketObs(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
```

```
        }

    S3Client { region = "us-east-1" }.use { s3 ->

        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The owner is ${myObject.owner}")
        }
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }
```

```
    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin 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

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CopyObject」を参照してください。

`CreateBucket`

次の例は、`CreateBucket` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createNewBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateBucket」を参照してください。

`CreateMultiRegionAccessPoint`

次の例は、`CreateMultiRegionAccessPoint` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

us-west-2 リージョンにリクエストを送信するように S3 コントロールクライアントを設定します。

```
        suspend fun createS3ControlClient(): S3ControlClient {
            // Configure your S3ControlClient to send requests to US West (Oregon).
            val s3Control = S3ControlClient.fromEnvironment {
                region = "us-west-2"
            }
            return s3Control
        }
```

マルチリージョンアクセスポイントを作成します。

```
    suspend fun createMrap(
        s3Control: S3ControlClient,
        accountIdParam: String,
        bucketName1: String,
        bucketName2: String,
        mrapName: String,
    ): String {
        println("Creating MRAP ...")
        val createMrapResponse: CreateMultiRegionAccessPointResponse =
            s3Control.createMultiRegionAccessPoint {
                accountId = accountIdParam
                clientToken = UUID.randomUUID().toString()
                details {
                    name = mrapName
                    regions = listOf(
                        Region {
                            bucket = bucketName1
                        },
                        Region {
                            bucket = bucketName2
                        },
                    )
                }
            }
        val requestToken: String? = createMrapResponse.requestTokenArn

        // Use the request token to check for the status of the
 CreateMultiRegionAccessPoint operation.
        if (requestToken != null) {
```

```
            waitForSucceededStatus(s3Control, requestToken, accountIdParam)
            println("MRAP created")
        }

        val getMrapResponse =
            s3Control.getMultiRegionAccessPoint(
                input = GetMultiRegionAccessPointRequest {
                    accountId = accountIdParam
                    name = mrapName
                },
            )
        val mrapAlias = getMrapResponse.accessPoint?.alias
        return "arn:aws:s3::$accountIdParam:accesspoint/$mrapAlias"
    }
```

マルチリージョンアクセスポイントが使用可能になるまで待ちます。

```
        suspend fun waitForSucceededStatus(
            s3Control: S3ControlClient,
            requestToken: String,
            accountIdParam: String,
            timeBetweenChecks: Duration = 1.minutes,
        ) {
            var describeResponse: DescribeMultiRegionAccessPointOperationResponse
            describeResponse = s3Control.describeMultiRegionAccessPointOperation(
                input = DescribeMultiRegionAccessPointOperationRequest {
                    accountId = accountIdParam
                    requestTokenArn = requestToken
                },
            )

            var status: String? = describeResponse.asyncOperation?.requestStatus
            while (status != "SUCCEEDED") {
                delay(timeBetweenChecks)
                describeResponse =
s3Control.describeMultiRegionAccessPointOperation(
                    input = DescribeMultiRegionAccessPointOperationRequest {
                        accountId = accountIdParam
                        requestTokenArn = requestToken
                    },
                )
                status = describeResponse.asyncOperation?.requestStatus
```

```
                println(status)
            }
        }
```

- 詳細については、「AWS SDK for Kotlin 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 Kotlin API リファレンス」の「CreateMultiRegionAccessPoint」を参照してください。

`DeleteBucketPolicy`

次の例は、`DeleteBucketPolicy` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteBucketPolicy」を参照してください。

`DeleteObjects`

次の例は、`DeleteObjects` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteBucketObjects(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DeleteObjects」を参照してください。

`GetBucketPolicy`

次の例は、`GetBucketPolicy` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetBucketPolicy」を参照してください。

`GetObject`

次の例は、`GetObject` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getObjectBytes(
    bucketName: String,
```

```
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetObject」を参照してください。

`GetObjectAcl`

次の例は、`GetObjectAcl` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
```

```
            key = objectKey
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetObjectAcl」を参照してください。

`ListObjectsV2`

次の例は、`ListObjectsV2` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${myObject.size?.let { calKb(it) }} KBs")
            println("The owner is ${myObject.owner}")
        }
    }
```

```
}

private fun calKb(intValue: Long): Long = intValue / 1024
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ListObjectsV2」を参照してください。

`PutBucketAcl`

次の例は、PutBucketAcl を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun setBucketAcl(
    bucketName: String,
    idVal: String,
) {
    val myGrant =
        Grantee {
            id = idVal
            type = Type.CanonicalUser
        }

    val ownerGrant =
        Grant {
            grantee = myGrant
            permission = Permission.FullControl
        }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb =
        Owner {
```

```
            id = idVal
        }

    val acl =
        AccessControlPolicy {
            owner = ownerOb
            grants = grantList
        }

    val request =
        PutBucketAclRequest {
            bucket = bucketName
            accessControlPolicy = acl
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「PutBucketAcl」を参照してください。

`PutObject`

次の例は、`PutObject` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
```

```
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            body = File(objectPath).asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「PutObject」を参照してください。

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

`GetObject` の署名済みリクエストを作成し、その URL を使用してオブジェクトをダウンロードします。

```
suspend fun getObjectPresigned(
```

```
    s3: S3Client,
    bucketName: String,
    keyName: String,
): String {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)

    // Use the URL from the presigned HttpRequest in a subsequent HTTP GET request
 to retrieve the object.
    val objectContents = URL(presignedRequest.url.toString()).readText()

    return objectContents
}
```

詳細オプションを使用して GetObject 署名済みリクエストを作成します。

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): HttpRequest {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
            signingDate = Instant.now() + 12.hours // Presigned request can be used
 12 hours from now.
            algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
            signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
```

```
            expiresAfter = 8.hours // Presigned request expires 8 hours later.
        }
    return presignedRequest
}
```

PutObject の署名済みリクエストを作成し、それを使用してオブジェクトをアップロードします。

```
suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
    val unsignedRequest =
        PutObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the request.
    val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)

    // Use the URL and any headers from the presigned HttpRequest in a subsequent
 HTTP PUT request to retrieve the object.
    // Create a PUT request using the OKHttpClient API.
    val putRequest =
        Request
            .Builder()
            .url(presignedRequest.url.toString())
            .apply {
                presignedRequest.headers.forEach { key, values ->
                    header(key, values.joinToString(", "))
                }
            }.put(content.toRequestBody())
            .build()

    val response = OkHttpClient().newCall(putRequest).execute()
    assert(response.isSuccessful)
}
```

- 詳細については、「AWS SDK for Kotlin デベロッパーガイド」を参照してください。

サーバーレスアプリケーションを作成して写真の管理

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

SDK for Kotlin

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設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、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

イメージ内のオブジェクトを検出する

次のコード例は、Amazon Rekognition を使用してイメージ内のカテゴリ別にオブジェクトを検出するアプリケーションを構築する方法を示しています。

SDK for Kotlin

Amazon Simple Storage Service (Amazon S3) バケットにある画像内からカテゴリ別にオブジェクトを Amazon Rekognition を使用して識別するアプリケーションを、Amazon Rekognition Kotlin API を使用して作成する方法を示します。アプリケーションは、Amazon Simple Email Service (Amazon SES) を使用して、結果を記載した 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および実行の手順については、GitHub で完全な例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 Rekognition
- Amazon S3
- Amazon SES

マルチリージョンアクセスポイント からオブジェクトを取得する

次のコード例は、マルチリージョンアクセスポイントからオブジェクトを取得する方法を示しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

非対称 Sigv4 (Sigv4a) 署名アルゴリズムを使用するように S3 クライアントを設定します。

```
        suspend fun createS3Client(): S3Client {
            // Configure your S3Client to use the Asymmetric SigV4 (SigV4a) signing
algorithm.
            val sigV4aScheme = SigV4AsymmetricAuthScheme(DefaultAwsSigner)
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4aScheme)
            }
            return s3
        }
```

バケット名の代わりにマルチリージョンアクセスポイント ARN を使用してオブジェクトを取得します。

```
    suspend fun getObjectFromMrap(
        s3: S3Client,
        mrapArn: String,
        keyName: String,
    ): String? {
```

```
        val request = GetObjectRequest {
            bucket = mrapArn // Use the ARN instead of the bucket name for object
operations.
            key = keyName
        }

        var stringObj: String? = null
        s3.getObject(request) { resp ->
            stringObj = resp.body?.decodeToString()
            if (stringObj != null) {
                println("Successfully read $keyName from $mrapArn")
            }
        }
        return stringObj
    }
```

- 詳細については、「AWS SDK for Kotlin 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 Kotlin API リファレンス」の「GetObject」を参照してください。

SDK for Kotlin を使用した SageMaker AI の例

次のコード例は、SageMaker AI で AWS SDK for Kotlin 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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

Hello SageMaker AI

次のコード例は、SageMaker AI の使用を開始する方法を示しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listBooks() {
    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
 sageMakerClient.listNotebookInstances(ListNotebookInstancesRequest {})
        response.notebookInstances?.forEach { item ->
            println("The notebook name is: ${item.notebookInstanceName}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ListNotebookInstances」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

`CreatePipeline`

次の例は、`CreatePipeline` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
 String?, pipelineNameVal: String?) {
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
        println(jsonObject)

        // Create the pipeline.
        val pipelineRequest = CreatePipelineRequest {
            pipelineDescription = "Kotlin SDK example pipeline"
            roleArn = roleArnVal
            pipelineName = pipelineNameVal
            pipelineDefinition = jsonObject.toString()
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            sageMakerClient.createPipeline(pipelineRequest)
        }
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスの「CreatePipeline」を参照してください。

`DeletePipeline`

次の例は、`DeletePipeline` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Delete a SageMaker pipeline by name.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.deletePipeline(pipelineRequest)
        println("*** Successfully deleted $pipelineNameVal")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DeletePipeline」を参照してください。

`DescribePipelineExecution`

次の例は、`DescribePipelineExecution` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
```

```
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
 sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribePipelineExecution」を参照してください。

`StartPipelineExecution`

次の例は、`StartPipelineExecution` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// Start a pipeline run with job configurations.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn: String?,
 pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
```

```
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
        value = roleArn
    }
    val para2 = Parameter {
        name = "parameter_queue_url"
        value = queueUrl
    }

    val inputJSON = """{
        "DataSourceConfig": {
        "S3Data": {
            "S3Uri": "s3://$bucketName/samplefiles/latlongtest.csv"
        },
        "Type": "S3_DATA"
        },
        "DocumentType": "CSV"
    }"""
    println(inputJSON)
    val para3 = Parameter {
        name = "parameter_vej_input_config"
        value = inputJSON
    }

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    val jobS3Data = VectorEnrichmentJobS3Data {
        s3Uri = output
    }

    val outputConfig = ExportVectorEnrichmentJobOutputConfig {
        s3Data = jobS3Data
    }

    val gson4: String = gson.toJson(outputConfig)
    val para4: Parameter = Parameter {
        name = "parameter_vej_export_config"
        value = gson4
    }
```

```
    println("parameter_vej_export_config:" + gson.toJson(outputConfig))

    val para5JSON =
        "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName\":
\"Longitude\",\"YAttributeName\":\"Latitude\"}}"

    val para5: Parameter = Parameter {
        name = "parameter_step_1_vej_config"
        value = para5JSON
    }

    parameters.add(para1)
    parameters.add(para2)
    parameters.add(para3)
    parameters.add(para4)
    parameters.add(para5)

    val pipelineExecutionRequest = StartPipelineExecutionRequest {
        pipelineExecutionDescription = "Created using Kotlin SDK"
        pipelineExecutionDisplayName = "$pipelineName-example-execution"
        pipelineParameters = parameters
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「StartPipelineExecution」を参照してください。

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 SDK を使用して SageMaker パイプラインを作成および実行する AWS SDKs」を参照してください。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
val DASHES = String(CharArray(80)).replace("\u0000", "-")
private var eventSourceMapping = ""

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
          <sageMakerRoleName> <lambdaRoleName> <functionName> <functionKey>
 <queueName> <bucketName> <bucketFunction> <lnglatData> <spatialPipelinePath>
 <pipelineName>

    Where:
        sageMakerRoleName - The name of the Amazon SageMaker role.
        lambdaRoleName - The name of the AWS Lambda role.
        functionName - The name of the AWS Lambda function (for
 example,SageMakerExampleFunction).
        functionKey - The name of the Amazon S3 key name that represents the Lambda
 function (for example, SageMakerLambda.zip).
        queueName - The name of the Amazon Simple Queue Service (Amazon SQS) queue.
        bucketName - The name of the Amazon Simple Storage Service (Amazon S3)
 bucket.
        bucketFunction - The name of the Amazon S3 bucket that contains the Lambda
 ZIP file.
```

```
        lnglatData - The file location of the latlongtest.csv file required for this
 use case.
        spatialPipelinePath - The file location of the GeoSpatialPipeline.json file
 required for this use case.
        pipelineName - The name of the pipeline to create (for example, sagemaker-
sdk-example-pipeline).
    """

    if (args.size != 10) {
        println(usage)
        exitProcess(1)
    }

    val sageMakerRoleName = args[0]
    val lambdaRoleName = args[1]
    val functionKey = args[2]
    val functionName = args[3]
    val queueName = args[4]
    val bucketName = args[5]
    val bucketFunction = args[6]
    val lnglatData = args[7]
    val spatialPipelinePath = args[8]
    val pipelineName = args[9]
    val handlerName = "org.example.SageMakerLambdaFunction::handleRequest"

    println(DASHES)
    println("Welcome to the Amazon SageMaker pipeline example scenario.")
    println(
        """
         This example workflow will guide you through setting up and running an
         Amazon SageMaker pipeline. The pipeline uses an AWS Lambda function and an
         Amazon SQS Queue. It runs a vector enrichment reverse geocode job to
         reverse geocode addresses in an input file and store the results in an
 export file.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println("First, we will set up the roles, functions, and queue needed by the
 SageMaker pipeline.")
    val lambdaRoleArn: String = checkLambdaRole(lambdaRoleName)
    val sageMakerRoleArn: String = checkSageMakerRole(sageMakerRoleName)
```

```
    val functionArn = checkFunction(functionName, bucketFunction, functionKey,
handlerName, lambdaRoleArn)
    val queueUrl = checkQueue(queueName, functionName)
    println(DASHES)

    println(DASHES)
    println("Setting up bucket $bucketName")
    if (!checkBucket(bucketName)) {
        setupBucket(bucketName)
        println("Put $lnglatData into $bucketName")
        val objectKey = "samplefiles/latlongtest.csv"
        putS3Object(bucketName, objectKey, lnglatData)
    }
    println(DASHES)

    println(DASHES)
    println("Now we can create and run our pipeline.")
    setupPipeline(spatialPipelinePath, sageMakerRoleArn, functionArn, pipelineName)
    val pipelineExecutionARN = executePipeline(bucketName, queueUrl,
sageMakerRoleArn, pipelineName)
    println("The pipeline execution ARN value is $pipelineExecutionARN")
    waitForPipelineExecution(pipelineExecutionARN)
    println("Wait 30 secs to get output results $bucketName")
    TimeUnit.SECONDS.sleep(30)
    getOutputResults(bucketName)
    println(DASHES)

    println(DASHES)
    println(
        """
            The pipeline has completed. To view the pipeline and runs in SageMaker
Studio, follow these instructions:
            https://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-studio.html
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println("Do you want to delete the AWS resources used in this Workflow? (y/n)")
    val `in` = Scanner(System.`in`)
    val delResources = `in`.nextLine()
    if (delResources.compareTo("y") == 0) {
        println("Lets clean up the AWS resources. Wait 30 seconds")
        TimeUnit.SECONDS.sleep(30)
```

```
        deleteEventSourceMapping(functionName)
        deleteSQSQueue(queueName)
        listBucketObjects(bucketName)
        deleteBucket(bucketName)
        delLambdaFunction(functionName)
        deleteLambdaRole(lambdaRoleName)
        deleteSagemakerRole(sageMakerRoleName)
        deletePipeline(pipelineName)
    } else {
        println("The AWS Resources were not deleted!")
    }
    println(DASHES)

    println(DASHES)
    println("SageMaker pipeline scenario is complete.")
    println(DASHES)
}

// Delete a SageMaker pipeline by name.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.deletePipeline(pipelineRequest)
        println("*** Successfully deleted $pipelineNameVal")
    }
}

suspend fun deleteSagemakerRole(roleNameVal: String) {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    IamClient { region = "us-west-2" }.use { iam ->
        for (policy in sageMakerRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
```

```
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteLambdaRole(roleNameVal: String) {
    val lambdaRolePolicies = getLambdaRolePolicies()
    IamClient { region = "us-west-2" }.use { iam ->
        for (policy in lambdaRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun delLambdaFunction(myFunctionName: String) {
    val request = DeleteFunctionRequest {
        functionName = myFunctionName
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request = DeleteBucketRequest {
        bucket = bucketName
    }
    S3Client { region = "us-east-1" }.use { s3 ->
```

```
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}

suspend fun deleteBucketObjects(bucketName: String, objectName: String?) {
    val toDelete = ArrayList<ObjectIdentifier>()
    val obId = ObjectIdentifier {
        key = objectName
    }
    toDelete.add(obId)
    val delOb = Delete {
        objects = toDelete
    }
    val dor = DeleteObjectsRequest {
        bucket = bucketName
        delete = delOb
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.deleteObjects(dor)
        println("*** $bucketName objects were deleted.")
    }
}

suspend fun listBucketObjects(bucketNameVal: String) {
    val listObjects = ListObjectsRequest {
        bucket = bucketNameVal
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val objects = res.contents
        if (objects != null) {
            for (myValue in objects) {
                println("The name of the key is ${myValue.key}")
                deleteBucketObjects(bucketNameVal, myValue.key)
            }
        }
    }
}

// Delete the specific Amazon SQS queue.
suspend fun deleteSQSQueue(queueNameVal: String?) {
```

```
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val urlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = urlVal
        }
        sqsClient.deleteQueue(deleteQueueRequest)
    }
}

// Delete the queue event mapping.
suspend fun deleteEventSourceMapping(functionNameVal: String) {
    if (eventSourceMapping.compareTo("") == 0) {
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
            val request = ListEventSourceMappingsRequest {
                functionName = functionNameVal
            }
            val response = lambdaClient.listEventSourceMappings(request)
            val eventList = response.eventSourceMappings
            if (eventList != null) {
                for (event in eventList) {
                    eventSourceMapping = event.uuid.toString()
                }
            }
        }
    }

    val eventSourceMappingRequest = DeleteEventSourceMappingRequest {
        uuid = eventSourceMapping
    }
    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest)
        println("The event mapping is deleted!")
    }
}

// Reads the objects in the S3 bucket and displays the values.
private suspend fun readObject(bucketName: String, keyVal: String?) {
    println("Output file contents: \n")
    val objectRequest = GetObjectRequest {
        bucket = bucketName
```

```
        key = keyVal
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.getObject(objectRequest) { resp ->
            val byteArray = resp.body?.toByteArray()
            val text = byteArray?.let { String(it, StandardCharsets.UTF_8) }
            println("Text output: $text")
        }
    }
}

// Display the results from the output directory.
suspend fun getOutputResults(bucketName: String?) {
    println("Getting output results $bucketName.")
    val listObjectsRequest = ListObjectsRequest {
        bucket = bucketName
        prefix = "outputfiles/"
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        val response = s3Client.listObjects(listObjectsRequest)
        val s3Objects: List<Object>? = response.contents
        if (s3Objects != null) {
            for (`object` in s3Objects) {
                if (bucketName != null) {
                    readObject(bucketName, (`object`.key))
                }
            }
        }
    }
}

suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
 sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
```

```
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}

// Start a pipeline run with job configurations.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn: String?,
 pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
        value = roleArn
    }
    val para2 = Parameter {
        name = "parameter_queue_url"
        value = queueUrl
    }

    val inputJSON = """{
        "DataSourceConfig": {
        "S3Data": {
            "S3Uri": "s3://$bucketName/samplefiles/latlongtest.csv"
        },
        "Type": "S3_DATA"
        },
        "DocumentType": "CSV"
    }"""
    println(inputJSON)
    val para3 = Parameter {
        name = "parameter_vej_input_config"
        value = inputJSON
```

```
    }

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    val jobS3Data = VectorEnrichmentJobS3Data {
        s3Uri = output
    }

    val outputConfig = ExportVectorEnrichmentJobOutputConfig {
        s3Data = jobS3Data
    }

    val gson4: String = gson.toJson(outputConfig)
    val para4: Parameter = Parameter {
        name = "parameter_vej_export_config"
        value = gson4
    }
    println("parameter_vej_export_config:" + gson.toJson(outputConfig))

    val para5JSON =
        "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName\":
\"Longitude\",\"YAttributeName\":\"Latitude\"}}"

    val para5: Parameter = Parameter {
        name = "parameter_step_1_vej_config"
        value = para5JSON
    }

    parameters.add(para1)
    parameters.add(para2)
    parameters.add(para3)
    parameters.add(para4)
    parameters.add(para5)

    val pipelineExecutionRequest = StartPipelineExecutionRequest {
        pipelineExecutionDescription = "Created using Kotlin SDK"
        pipelineExecutionDisplayName = "$pipelineName-example-execution"
        pipelineParameters = parameters
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
```

```
    }
}

// Create a pipeline from the example pipeline JSON.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
 String?, pipelineNameVal: String?) {
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
        println(jsonObject)

        // Create the pipeline.
        val pipelineRequest = CreatePipelineRequest {
            pipelineDescription = "Kotlin SDK example pipeline"
            roleArn = roleArnVal
            pipelineName = pipelineNameVal
            pipelineDefinition = jsonObject.toString()
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            sageMakerClient.createPipeline(pipelineRequest)
        }
    }
}

suspend fun putS3Object(bucketName: String, objectKey: String, objectPath: String) {
    val request = PutObjectRequest {
        bucket = bucketName
        key = objectKey
        body = File(objectPath).asByteStream()
    }

    S3Client { region = "us-east-1" }.use { s3 ->
```

```
        s3.putObject(request)
        println("Successfully placed $objectKey into bucket $bucketName")
    }
}

suspend fun setupBucket(bucketName: String) {
    val request = CreateBucketRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun checkBucket(bucketName: String): Boolean {
    try {
        val headBucketRequest = HeadBucketRequest {
            bucket = bucketName
        }
        S3Client { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            println("$bucketName exists")
            return true
        }
    } catch (e: S3Exception) {
        println("Bucket does not exist")
    }
    return false
}

// Connect the queue to the Lambda function as an event source.
suspend fun connectLambda(queueUrlVal: String?, lambdaNameVal: String?) {
    println("Connecting the Lambda function and queue for the pipeline.")
    var queueArn = ""

    // Specify the attributes to retrieve.
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)
    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
```

```
    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val queueAtts = response.attributes
        if (queueAtts != null) {
            for ((key, value) in queueAtts) {
                println("Key = $key, Value = $value")
                queueArn = value
            }
        }
    }
    val eventSourceMappingRequest = CreateEventSourceMappingRequest {
        eventSourceArn = queueArn
        functionName = lambdaNameVal
    }
    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        val response1 =
 lambdaClient.createEventSourceMapping(eventSourceMappingRequest)
        eventSourceMapping = response1.uuid.toString()
        println("The mapping between the event source and Lambda function was
 successful")
    }
}

// Set up the SQS queue to use with the pipeline.
suspend fun setupQueue(queueNameVal: String, lambdaNameVal: String): String {
    println("Setting up queue named $queueNameVal")
    val queueAtt: MutableMap<String, String> = HashMap()
    queueAtt.put("DelaySeconds", "5")
    queueAtt.put("ReceiveMessageWaitTimeSeconds", "5")
    queueAtt.put("VisibilityTimeout", "300")

    val createQueueRequest = CreateQueueRequest {
        queueName = queueNameVal
        attributes = queueAtt
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("\nGet queue url")
        val getQueueUrlResponse = sqsClient.getQueueUrl(GetQueueUrlRequest
 { queueName = queueNameVal })
        TimeUnit.SECONDS.sleep(15)
        connectLambda(getQueueUrlResponse.queueUrl, lambdaNameVal)
```

```
        println("Queue ready with Url " + getQueueUrlResponse.queueUrl)
        return getQueueUrlResponse.queueUrl.toString()
    }
}

// Checks to see if the Amazon SQS queue exists. If not, this method creates a new
 queue
// and returns the ARN value.
suspend fun checkQueue(queueNameVal: String, lambdaNameVal: String): String? {
    println("Checking to see if the queue exists. If not, a new queue will be
 created for use in this workflow.")
    var queueUrl: String
    try {
        val request = GetQueueUrlRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-west-2" }.use { sqsClient ->
            val response = sqsClient.getQueueUrl(request)
            queueUrl = response.queueUrl.toString()
            println(queueUrl)
        }
    } catch (e: SqsException) {
        println(e.message + " A new queue will be created")
        queueUrl = setupQueue(queueNameVal, lambdaNameVal)
    }
    return queueUrl
}

suspend fun createNewFunction(myFunctionName: String, s3BucketName: String, myS3Key:
 String, myHandler: String, myRole: String): String {
    val functionCode = FunctionCode {
        s3Bucket = s3BucketName
        s3Key = myS3Key
    }

    val request = CreateFunctionRequest {
        functionName = myFunctionName
        code = functionCode
        description = "Created by the Lambda Kotlin API"
        handler = myHandler
        role = myRole
        runtime = Runtime.Java11
        memorySize = 1024
```

```
        timeout = 200
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        println("${functionResponse.functionArn} was created")
        return functionResponse.functionArn.toString()
    }
}

suspend fun checkFunction(myFunctionName: String, s3BucketName: String, myS3Key:
 String, myHandler: String, myRole: String): String {
    println("Checking to see if the function exists. If not, a new AWS Lambda
 function will be created for use in this workflow.")
    var functionArn: String
    try {
        // Does this function already exist.
        val functionRequest = GetFunctionRequest {
            functionName = myFunctionName
        }
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
            val response = lambdaClient.getFunction(functionRequest)
            functionArn = response.configuration?.functionArn.toString()
            println("$functionArn exists")
        }
    } catch (e: LambdaException) {
        println(e.message + " A new function will be created")
        functionArn = createNewFunction(myFunctionName, s3BucketName, myS3Key,
 myHandler, myRole)
    }
    return functionArn
}

// Checks to see if the SageMaker role exists. If not, this method creates it.
suspend fun checkSageMakerRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be created
 for AWS SageMaker to use.")
    var roleArn: String
    try {
        val roleRequest = GetRoleRequest {
            roleName = roleNameVal
```

```
        }
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
    } catch (e: IamException) {
        println(e.message + " A new role will be created")
        roleArn = createSageMakerRole(roleNameVal)
    }
    return roleArn
}

suspend fun createSageMakerRole(roleNameVal: String): String {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    println("Creating a role to use with SageMaker.")
    val 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\"" +
        "}]" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in sageMakerRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
```

```
                policyArn = policy
            }
            iamClient.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.toString()
    }
}

// Checks to see if the Lambda role exists. If not, this method creates it.
suspend fun checkLambdaRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be created
 for AWS Lambda to use.")
    var roleArn: String
    val roleRequest = GetRoleRequest {
        roleName = roleNameVal
    }

    try {
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
    } catch (e: IamException) {
        println(e.message + " A new role will be created")
        roleArn = createLambdaRole(roleNameVal)
    }

    return roleArn
}

private suspend fun createLambdaRole(roleNameVal: String): String {
    val lambdaRolePolicies = getLambdaRolePolicies()
    val 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\"" +
        "}]" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in lambdaRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policy
            }
            iamClient.attachRolePolicy(attachRequest)
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15)
        println("Role ready with ARN " + roleResult.role?.arn)
        return roleResult.role?.arn.toString()
    }
}

fun getLambdaRolePolicies(): Array<String?> {
    val lambdaRolePolicies = arrayOfNulls<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
}

fun getSageMakerRolePolicies(): Array<String?> {
    val sageMakerRolePolicies = arrayOfNulls<String>(3)
    sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess"
    sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess"
    sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess"
    return sageMakerRolePolicies
}
```

- API の詳細については、「AWS SDK for Kotlin 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 Kotlin を使用した Secrets Manager の例

次のコード例は、Secrets Manager で AWS SDK for Kotlin 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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getValue(secretName: String?) {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        val response = secretsClient.getSecretValue(valueRequest)
        val secret = response.secretString
        println("The secret value is $secret")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「GetSecretValue」を参照してください。

SDK for Kotlin を使用した Amazon SES の例

次のコード例は、Amazon SES で AWS SDK for Kotlin 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。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- シナリオ

シナリオ

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

次のコード例は、Amazon DynamoDB テーブル内の作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信する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。

SDK for Kotlin

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および実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- DynamoDB
- Amazon SES

Amazon Redshift データ追跡用のウェブアプリケーションの作成

次のコード例は、Amazon Redshift データベースを使用して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示しています。

SDK for Kotlin

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 サーバーレスデータをクエリする Spring REST API と React アプリケーションで使用するための完全なソースコードと設定方法については、GitHub にある詳しい事例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 Redshift
- Amazon SES

Aurora Serverless 作業項目トラッカーの作成

次のコード例は、Amazon Aurora Serverless データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信する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。

SDK for Kotlin

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 サーバーレスデータをクエリする Spring REST API と React アプリケーションで使用するための完全なソースコードと設定方法については、GitHub にある完全な例を参照してください。

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

イメージ内のオブジェクトを検出する

次のコード例は、Amazon Rekognition を使用してイメージ内のカテゴリ別にオブジェクトを検出するアプリケーションを構築する方法を示しています。

SDK for Kotlin

Amazon Simple Storage Service (Amazon S3) バケットにある画像内からカテゴリ別にオブジェクトを Amazon Rekognition を使用して識別するアプリケーションを、Amazon Rekognition Kotlin API を使用して作成する方法を示します。アプリケーションは、Amazon Simple Email Service (Amazon SES) を使用して、結果を記載した 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および実行の手順については、GitHub で完全な例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 Rekognition
- Amazon S3
- Amazon SES

SDK for Kotlin を使用した Amazon SNS の例

次のコード例は、Amazon SNS で AWS SDK for Kotlin 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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import aws.sdk.kotlin.services.sns.SnsClient
```

```
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ListTopics」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

CreateTopic

次の例は、CreateTopic を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateTopic」を参照してください。

`DeleteTopic`

次の例は、`DeleteTopic` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
```

```
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DeleteTopic」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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「GetTopicAttributes」を参照してください。

`ListSubscriptions`

次の例は、`ListSubscriptions` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listSNSSubscriptions() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ListSubscriptions」を参照してください。

`ListTopics`

次の例は、`ListTopics` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listSNSTopics() {
```

```
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ListTopics」を参照してください。

`Publish`

次の例は、`Publish` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun pubTopic(
    topicArnVal: String,
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「Publish」を参照してください。

SetTopicAttributes

次の例は、SetTopicAttributes を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「SetTopicAttributes」を参照してください。

Subscribe

次の例は、Subscribe を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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。

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

Lambda 関数をトピックにサブスクライブします。

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }
```

```
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin 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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun addTopicTags(topicArn: String) {
    val tag =
        Tag {
            key = "Team"
            value = "Development"
        }

    val tag2 =
        Tag {
            key = "Environment"
            value = "Gamma"
        }

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)

    val request =
```

```
        TagResourceRequest {
            resourceArn = topicArn
            tags = tagList
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「TagResource」を参照してください。

`Unsubscribe`

次の例は、`Unsubscribe` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「Unsubscribe」を参照してください。

シナリオ

Amazon SNS アプリケーションの構築

次のコード例は、サブスクリプションとパブリッシュ機能を持ち、メッセージを翻訳するアプリケーションを作成する方法を示しています。

SDK for Kotlin

Amazon SNS Kotlin API を使用して、サブスクリプションおよび発行機能を持つアプリケーションを作成する方法を示しています。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。

詳細なソースコードとウェブアプリケーションを作成する手順については、「GitHub」の詳しい例を参照してください。

詳細なソースコードと Android ネイティブアプリケーションを作成する手順については、「GitHub」の詳しい例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 SNS
- Amazon Translate

サーバーレスアプリケーションを作成して写真の管理

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

SDK for Kotlin

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設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、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

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「Publish」を参照してください。

メッセージをキューに発行する

次のコードサンプルは、以下の操作方法を示しています。

- トピック (FIFO または非 FIFO) を作成します。
- フィルターを適用するオプションを使用して、複数のキューをトピックにサブスクライブします。
- メッセージをトピックに発行します。
- キューをポーリングして受信メッセージを確認します。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
package com.example.sns

import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
```

```
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
```

```
    println("Welcome to the AWS SDK for Kotlin messaging with topics and queues.")
    println(
        """
                In this scenario, you will create an SNS topic and subscribe an SQS
 queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication and
 message filtering.
                Would you like to work with FIFO topics? (y/n)
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
    if (useFIFO.compareTo("y") == 0) {
        selectFIFO = true
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically generated
 from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
 delivered.
        For more information about deduplication, see https://docs.aws.amazon.com/
sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
```

```
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
            groupId = input.nextLine()
        }
    }
    println(DASHES)

    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended to
the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
```

```
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
         },
         "Action": "sqs:SendMessage",
                 "Resource": "$sqsQueueArn",
                 "Condition": {
             "ArnEquals": {
                 "aws:SourceArn": "$topicArn"
             }
         }
     }
     ]
     }"""
    setQueueAttr(sqsQueueUrl, policy)
    println(DASHES)

    println(DASHES)
    println("6. Subscribe to the SQS queue.")
    if (selectFIFO) {
        println(
            """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/latest/
dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
        )
        println("Would you like to filter messages for $sqsQueueName's subscription
 to the topic $topicName?  (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the following
 \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
```

```
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
                    else -> moreAns = true
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following \"tone
\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
 deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
```

```
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key to
 continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
```

```
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was deleted")
    }
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }
```

```
    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
```

```
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
 aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
```

```
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
```

```
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic " +
 topicArnVal + "\n" +
                     "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}
```

```
suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
```

```
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}
```

```
suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- API の詳細については、『AWS SDK for Kotlin API リファレンス』の以下のトピックを参照してください。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - 公開
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

SDK for Kotlin を使用した Amazon SQS の例

次のコード例は、Amazon SQS で AWS SDK for Kotlin 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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
package com.kotlin.sqs

import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.paginators.listQueuesPaginated
import kotlinx.coroutines.flow.transform

suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient
```

```
            .listQueuesPaginated { }
            .transform { it.queueUrls?.forEach { queue -> emit(queue) } }
            .collect { queue ->
                println("The Queue URL is $queue")
            }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ListQueues」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

`CreateQueue`

次の例は、`CreateQueue` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createQueue(queueNameVal: String): String {
    println("Create Queue")
    val createQueueRequest =
        CreateQueueRequest {
            queueName = queueNameVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("Get queue url")
```

```
        val getQueueUrlRequest =
            GetQueueUrlRequest {
                queueName = queueNameVal
            }

        val getQueueUrlResponse = sqsClient.getQueueUrl(getQueueUrlRequest)
        return getQueueUrlResponse.queueUrl.toString()
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CreateQueue」を参照してください。

`DeleteMessage`

次の例は、`DeleteMessage` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}
```

```
suspend fun deleteQueue(queueUrlVal: String) {
    val request =
        DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DeleteMessage」を参照してください。

`DeleteQueue`

次の例は、`DeleteQueue` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}
```

```
suspend fun deleteQueue(queueUrlVal: String) {
    val request =
        DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DeleteQueue」を参照してください。

`ListQueues`

次の例は、`ListQueues` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listQueues() {
    println("\nList Queues")

    val prefix = "que"
    val listQueuesRequest =
        ListQueuesRequest {
            queueNamePrefix = prefix
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.listQueues(listQueuesRequest)
        response.queueUrls?.forEach { url ->
```

```
            println(url)
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ListQueues」を参照してください。

`ReceiveMessage`

次の例は、`ReceiveMessage` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun receiveMessages(queueUrlVal: String?) {
    println("Retrieving messages from $queueUrlVal")

    val receiveMessageRequest =
        ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.receiveMessage(receiveMessageRequest)
        response.messages?.forEach { message ->
            println(message.body)
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ReceiveMessage」を参照してください。

SendMessage

次の例は、SendMessage を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun sendMessages(
    queueUrlVal: String,
    message: String,
) {
    println("Sending multiple messages")
    println("\nSend message")
    val sendRequest =
        SendMessageRequest {
            queueUrl = queueUrlVal
            messageBody = message
            delaySeconds = 10
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessage(sendRequest)
        println("A single message was successfully sent.")
    }
}

suspend fun sendBatchMessages(queueUrlVal: String?) {
    println("Sending multiple messages")

    val msg1 =
        SendMessageBatchRequestEntry {
            id = "id1"
            messageBody = "Hello from msg 1"
        }

    val msg2 =
        SendMessageBatchRequestEntry {
            id = "id2"
```

```
            messageBody = "Hello from msg 2"
        }

    val sendMessageBatchRequest =
        SendMessageBatchRequest {
            queueUrl = queueUrlVal
            entries = listOf(msg1, msg2)
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessageBatch(sendMessageBatchRequest)
        println("Batch message were successfully sent.")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「SendMessage」を参照してください。

シナリオ

Slack メッセージングアプリケーションを作成します。

次のコード例は、Amazon SQS を使用してメッセージングアプリケーションを作成する方法を示しています。

SDK for Kotlin

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および実行の手順については、GitHub で完全な例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
- Amazon SQS

メッセージをキューに発行する

次のコードサンプルは、以下の操作方法を示しています。

- トピック (FIFO または非 FIFO) を作成します。
- フィルターを適用するオプションを使用して、複数のキューをトピックにサブスクライブします。
- メッセージをトピックに発行します。
- キューをポーリングして受信メッセージを確認します。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
package com.example.sns

import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
```

```
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and queues.")
    println(
        """
```

```
                In this scenario, you will create an SNS topic and subscribe an SQS
 queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
            """.trimIndent(),
        )
        println(DASHES)

        println(DASHES)
        println(
            """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication and
 message filtering.
                Would you like to work with FIFO topics? (y/n)
            """.trimIndent(),
        )
        useFIFO = input.nextLine()
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true
            println("You have selected FIFO")
            println(
                """ Because you have chosen a FIFO topic, deduplication is supported.
            Deduplication IDs are either set in the message or automatically generated
 from content using a hash function.
            If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
            within the five-minute deduplication interval, is accepted but not
 delivered.
            For more information about deduplication, see https://docs.aws.amazon.com/
sns/latest/dg/fifo-message-dedup.html.""",
            )

            println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
            duplication = input.nextLine()
            if (duplication.compareTo("y") == 0) {
                println("Enter a group id value")
                groupId = input.nextLine()
            } else {
                println("Enter deduplication Id value")
                deduplicationID = input.nextLine()
                println("Enter a group id value")
```

```
            groupId = input.nextLine()
        }
    }
    println(DASHES)

    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended to
the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
```

```
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
         },
         "Action": "sqs:SendMessage",
                 "Resource": "$sqsQueueArn",
                 "Condition": {
             "ArnEquals": {
                 "aws:SourceArn": "$topicArn"
             }
         }
     }
     ]
     }"""
    setQueueAttr(sqsQueueUrl, policy)
    println(DASHES)

    println(DASHES)
    println("6. Subscribe to the SQS queue.")
    if (selectFIFO) {
        println(
            """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/latest/
dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
        )
        println("Would you like to filter messages for $sqsQueueName's subscription
 to the topic $topicName?  (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the following
 \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
```

```
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
                    else -> moreAns = true
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following \"tone
\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
 deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
        pubMessage(message, topicArn)
    }
    println(DASHES)
```

```
    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key to
 continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }
```

```
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was deleted")
    }
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
```

```
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}

suspend fun pubMessageFIFO(
```

```
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
 aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
```

```
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
```

```
                "The queue " + queueArnVal + " has been subscribed to the topic " +
 topicArnVal + "\n" +
                    "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
```

```
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }
```

```
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
```

```
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- API の詳細については、『AWS SDK for Kotlin API リファレンス』の以下のトピックを参照してください。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - 公開
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

SDK for Kotlin を使用した Step Function の例

次のコード例は、Step Functions で AWS SDK for Kotlin 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。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest

/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

suspend fun main() {
    println(DASHES)
    println("Welcome to the AWS Step Functions Hello example.")
    println("Lets list up to ten of your state machines:")
    println(DASHES)

    listMachines()
}
```

```
suspend fun listMachines() {
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ListStateMachines」を参照してください。

トピック

- 基本
- アクション

基本

基本を学ぶ

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import aws.sdk.kotlin.services.iam.IamClient
import aws.sdk.kotlin.services.iam.model.CreateRoleRequest
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.CreateActivityRequest
import aws.sdk.kotlin.services.sfn.model.CreateStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DeleteActivityRequest
import aws.sdk.kotlin.services.sfn.model.DeleteStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DescribeExecutionRequest
import aws.sdk.kotlin.services.sfn.model.DescribeStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.GetActivityTaskRequest
import aws.sdk.kotlin.services.sfn.model.ListActivitiesRequest
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest
import aws.sdk.kotlin.services.sfn.model.SendTaskSuccessRequest
import aws.sdk.kotlin.services.sfn.model.StartExecutionRequest
import aws.sdk.kotlin.services.sfn.model.StateMachineType
import aws.sdk.kotlin.services.sfn.paginators.listActivitiesPaginated
import aws.sdk.kotlin.services.sfn.paginators.listStateMachinesPaginated
import com.fasterxml.jackson.databind.JsonNode
import com.fasterxml.jackson.databind.ObjectMapper
import com.fasterxml.jackson.databind.node.ObjectNode
import kotlinx.coroutines.flow.transform
import java.util.Scanner
import java.util.UUID
import kotlin.collections.ArrayList
import kotlin.system.exitProcess

/**
 To run this code example, place the chat_sfn_state_machine.json file into your
 project's resources folder.

 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

 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin code example performs the following tasks:
```

```
 1. List activities using a paginator.
 2. List state machines using a paginator.
 3. Creates an activity.
 4. Creates a state machine.
 5. Describes the state machine.
 6. Starts execution of the state machine and interacts with it.
 7. Describes the execution.
 8. Deletes the activity.
 9. Deletes the state machine.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <roleARN> <activityName> <stateMachineName>

    Where:
        roleName - The name of the IAM role to create for this state machine.
        activityName - The name of an activity to create.
        stateMachineName - The name of the state machine to create.
        jsonFile - The location of the chat_sfn_state_machine.json file. You can
 located it in resources/sample_files.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(0)
    }

    val roleName = args[0]
    val activityName = args[1]
    val stateMachineName = args[2]
    val jsonFile = args[3]
    val sc = Scanner(System.`in`)
    var action = false

    val polJSON = """{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
```

```
            "Principal": {
                "Service": "states.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
    }"""

    println(DASHES)
    println("Welcome to the AWS Step Functions example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List activities using a Paginator.")
    listActivitesPagnator()
    println(DASHES)

    println(DASHES)
    println("2. List state machines using a paginator.")
    listStatemachinesPagnator()
    println(DASHES)

    println(DASHES)
    println("3. Create a new activity.")
    val activityArn = createActivity(activityName)
    println("The ARN of the Activity is $activityArn")
    println(DASHES)

    // Get JSON to use for the state machine and place the activityArn value into
it.
    val stream = GetStream()
    val jsonString = stream.getStream(jsonFile)

    // Modify the Resource node.
    val objectMapper = ObjectMapper()
    val root: JsonNode = objectMapper.readTree(jsonString)
    (root.path("States").path("GetInput") as ObjectNode).put("Resource",
activityArn)

    // Convert the modified Java object back to a JSON string.
    val stateDefinition = objectMapper.writeValueAsString(root)
    println(stateDefinition)

    println(DASHES)
```

```
    println("4. Create a state machine.")
    val roleARN = createIAMRole(roleName, polJSON)
    val stateMachineArn = createMachine(roleARN, stateMachineName, stateDefinition)
    println("The ARN of the state machine is $stateMachineArn")
    println(DASHES)

    println(DASHES)
    println("5. Describe the state machine.")
    describeStateMachine(stateMachineArn)
    println("What should ChatSFN call you?")
    val userName = sc.nextLine()
    println("Hello $userName")
    println(DASHES)

    println(DASHES)
    // The JSON to pass to the StartExecution call.
    val executionJson = "{ \"name\" : \"$userName\" }"
    println(executionJson)
    println("6. Start execution of the state machine and interact with it.")
    val runArn = startWorkflow(stateMachineArn, executionJson)
    println("The ARN of the state machine execution is $runArn")
    var myList: List<String>
    while (!action) {
        myList = getActivityTask(activityArn)
        println("ChatSFN: " + myList[1])
        println("$userName please specify a value.")
        val myAction = sc.nextLine()
        if (myAction.compareTo("done") == 0) {
            action = true
        }
        println("You have selected $myAction")
        val taskJson = "{ \"action\" : \"$myAction\" }"
        println(taskJson)
        sendTaskSuccess(myList[0], taskJson)
    }
    println(DASHES)

    println(DASHES)
    println("7. Describe the execution.")
    describeExe(runArn)
    println(DASHES)

    println(DASHES)
    println("8. Delete the activity.")
```

```
    deleteActivity(activityArn)
    println(DASHES)

    println(DASHES)
    println("9. Delete the state machines.")
    deleteMachine(stateMachineArn)
    println(DASHES)

    println(DASHES)
    println("The AWS Step Functions example scenario is complete.")
    println(DASHES)
}

suspend fun listStatemachinesPagnator() {
    val machineRequest =
        ListStateMachinesRequest {
            maxResults = 10
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient
            .listStateMachinesPaginated(machineRequest)
            .transform { it.stateMachines?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" The state machine ARN is ${obj.stateMachineArn}")
            }
    }
}

suspend fun listActivitesPagnator() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient
            .listActivitiesPaginated(activitiesRequest)
            .transform { it.activities?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" The activity ARN is ${obj.activityArn}")
            }
    }
}
```

```
suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}

suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}

suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient { region = "us-east-1" }.use { sfnClient ->
            val response = sfnClient.describeExecution(executionRequest)
            status = response.status.toString()
            if (status.compareTo("Running") == 0) {
                println("The state machine is still running, let's wait for it to
 finish.")
                Thread.sleep(2000)
            } else if (status.compareTo("Succeeded") == 0) {
                println("The Step Function workflow has succeeded")
                hasSucceeded = true
```

```
            } else {
                println("The Status is $status")
            }
        }
    }
    println("The Status is $status")
}

suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
            output = json
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}

suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
}

suspend fun startWorkflow(
    stateMachineArnVal: String?,
    jsonEx: String?,
): String? {
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val executionRequest =
        StartExecutionRequest {
```

```
            input = jsonEx
            stateMachineArn = stateMachineArnVal
            name = uuidValue
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}

suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.describeStateMachine(stateMachineRequest)
        println("The name of the State machine is ${response.name}")
        println("The status of the State machine is ${response.status}")
        println("The ARN value of the State machine is ${response.stateMachineArn}")
        println("The role ARN value is ${response.roleArn}")
    }
}

suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}

suspend fun createIAMRole(
```

```
    roleNameVal: String?,
    polJSON: String?,
): String? {
    val request =
        CreateRoleRequest {
            roleName = roleNameVal
            assumeRolePolicyDocument = polJSON
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- API の詳細については、「AWS SDK for Kotlin 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

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CreateActivity」を参照してください。

`CreateStateMachine`

次の例は、`CreateStateMachine` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CreateStateMachine」を参照してください。

`DeleteActivity`

次の例は、`DeleteActivity` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DeleteActivity」を参照してください。

`DeleteStateMachine`

次の例は、`DeleteStateMachine` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
```

```
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DeleteStateMachine」を参照してください。

`DescribeExecution`

次の例は、`DescribeExecution` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient { region = "us-east-1" }.use { sfnClient ->
            val response = sfnClient.describeExecution(executionRequest)
            status = response.status.toString()
            if (status.compareTo("Running") == 0) {
                println("The state machine is still running, let's wait for it to
 finish.")
                Thread.sleep(2000)
            } else if (status.compareTo("Succeeded") == 0) {
```

```
                println("The Step Function workflow has succeeded")
                hasSucceeded = true
            } else {
                println("The Status is $status")
            }
        }
    }
    println("The Status is $status")
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeExecution」を参照してください。

`DescribeStateMachine`

次の例は、`DescribeStateMachine` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.describeStateMachine(stateMachineRequest)
        println("The name of the State machine is ${response.name}")
        println("The status of the State machine is ${response.status}")
        println("The ARN value of the State machine is ${response.stateMachineArn}")
        println("The role ARN value is ${response.roleArn}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeStateMachine」を参照してください。

`GetActivityTask`

次の例は、`GetActivityTask` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「GetActivityTask」を参照してください。

`ListActivities`

次の例は、`ListActivities` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listAllActivites() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listActivities(activitiesRequest)
        response.activities?.forEach { item ->
            println("The activity ARN is ${item.activityArn}")
            println("The activity name is ${item.name}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ListActivities」を参照してください。

`ListExecutions`

次の例は、`ListExecutions` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getExeHistory(exeARN: String?) {
```

```
    val historyRequest =
        GetExecutionHistoryRequest {
            executionArn = exeARN
            maxResults = 10
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getExecutionHistory(historyRequest)
        response.events?.forEach { event ->
            println("The event type is ${event.type}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ListExecutions」を参照してください。

`ListStateMachines`

次の例は、`ListStateMachines` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest

/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
```

```
suspend fun main() {
    println(DASHES)
    println("Welcome to the AWS Step Functions Hello example.")
    println("Lets list up to ten of your state machines:")
    println(DASHES)

    listMachines()
}

suspend fun listMachines() {
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ListStateMachines」を参照してください。

`SendTaskSuccess`

次の例は、`SendTaskSuccess` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
```

```
            taskToken = token
            output = json
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「SendTaskSuccess」を参照してください。

`StartExecution`

次の例は、`StartExecution` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun startWorkflow(
    stateMachineArnVal: String?,
    jsonEx: String?,
): String? {
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val executionRequest =
        StartExecutionRequest {
            input = jsonEx
            stateMachineArn = stateMachineArnVal
            name = uuidValue
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「StartExecution」を参照してください。

サポート SDK for Kotlin を使用した の例

次のコード例は、 で AWS SDK for Kotlin 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 サポート。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

開始方法

こんにち サポートは

次のコード例は、 サポートの使用を開始する方法を示しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.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 Kotlin example performs the following task:

1. Gets and displays available services.
 */

suspend fun main() {
    displaySomeServices()
}

// Return a List that contains a Service name and Category name.
suspend fun displaySomeServices() {
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is: " + service.name)

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                index++
            }
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeServices」を参照してください。

トピック

- 基本
- アクション

基本

基本を学ぶ

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
```

```
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.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 Kotlin 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.
*/

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <fileAttachment>
    Where:
         fileAttachment - The file can be a simple saved .txt file to use as an
 email attachment.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val fileAttachment = args[0]
    println("***** Welcome to the AWS Support case example scenario.")
    println("***** Step 1. Get and display available services.")
    val sevCatList = displayServices()

    println("***** Step 2. Get and display Support severity levels.")
    val sevLevel = displaySevLevels()

    println("***** Step 3. Create a support case using the selected service,
 category, and severity level.")
```

```
    val caseIdVal = createSupportCase(sevCatList, sevLevel)
    if (caseIdVal != null) {
        println("Support case $caseIdVal was successfully created!")
    } else {
        println("A support case was not successfully created!")
        exitProcess(1)
    }

    println("***** Step 4. Get open support cases.")
    getOpenCase()

    println("***** Step 5. Create an attachment set with a generated file to add to
 the case.")
    val attachmentSetId = addAttachment(fileAttachment)
    println("The Attachment Set id value is $attachmentSetId")

    println("***** Step 6. Add communication with the attachment to the support
 case.")
    addAttachSupportCase(caseIdVal, attachmentSetId)

    println("***** Step 7. List the communications of the support case.")
    val attachId = listCommunications(caseIdVal)
    println("The Attachment id value is $attachId")

    println("***** Step 8. Describe the attachment set included with the
 communication.")
    describeAttachment(attachId)

    println("***** Step 9. Resolve the support case.")
    resolveSupportCase(caseIdVal)

    println("***** Step 10. Get a list of resolved cases for the current day.")
    getResolvedCase()
    println("***** This Scenario has successfully completed")
}

suspend fun getResolvedCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 30
```

```
            afterTime = yesterday.toString()
            beforeTime = now.toString()
            includeResolvedCases = true
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}

suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}

suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
            caseId = caseIdVal
            maxResults = 10
        }
```

```
    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}

suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS Support
 case")
        } else {
            println("There was an error adding the communication to an AWS Support
 case")
        }
    }
}

suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal =
        Attachment {
            fileName = myFile.name
            data = sourceBytes
        }
```

```
    val setRequest =
        AddAttachmentsToSetRequest {
            attachments = listOf(attachmentVal)
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}

suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 20
            afterTime = yesterday.toString()
            beforeTime = now.toString()
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun createSupportCase(
    sevCatListVal: List<String>,
    sevLevelVal: String,
): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest =
        CreateCaseRequest {
            categoryCode = caseCategory.lowercase(Locale.getDefault())
            serviceCode = serCode.lowercase(Locale.getDefault())
```

```
            severityCode = sevLevelVal.lowercase(Locale.getDefault())
            communicationBody = "Test issue with
 ${serCode.lowercase(Locale.getDefault())}"
            subject = "Test case, please ignore"
            language = "en"
            issueType = "technical"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}

suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
        return levelName
    }
}

// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
```

```
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- API の詳細については、「AWS SDK for Kotlin 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

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 Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal =
        Attachment {
            fileName = myFile.name
            data = sourceBytes
        }

    val setRequest =
        AddAttachmentsToSetRequest {
            attachments = listOf(attachmentVal)
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「AddAttachmentsToSet」を参照してください。

AddCommunicationToCase

次の例は、AddCommunicationToCase を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS Support
 case")
        } else {
            println("There was an error adding the communication to an AWS Support
 case")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「AddCommunicationToCase」を参照してください。

`CreateCase`

次の例は、`CreateCase` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createSupportCase(
    sevCatListVal: List<String>,
    sevLevelVal: String,
): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest =
        CreateCaseRequest {
            categoryCode = caseCategory.lowercase(Locale.getDefault())
            serviceCode = serCode.lowercase(Locale.getDefault())
            severityCode = sevLevelVal.lowercase(Locale.getDefault())
            communicationBody = "Test issue with
 ${serCode.lowercase(Locale.getDefault())}"
            subject = "Test case, please ignore"
            language = "en"
            issueType = "technical"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「CreateCase」を参照してください。

`DescribeAttachment`

次の例は、`DescribeAttachment` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeAttachment」を参照してください。

`DescribeCases`

次の例は、`DescribeCases` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 20
            afterTime = yesterday.toString()
            beforeTime = now.toString()
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeCases」を参照してください。

`DescribeCommunications`

次の例は、`DescribeCommunications` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
```

```
            caseId = caseIdVal
            maxResults = 10
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeCommunications」を参照してください。

`DescribeServices`

次の例は、`DescribeServices` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、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.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }
```

```
    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeServices」を参照してください。

`DescribeSeverityLevels`

次の例は、`DescribeSeverityLevels` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
        return levelName
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「DescribeSeverityLevels」を参照してください。

`ResolveCase`

次の例は、`ResolveCase` を使用する方法を説明しています。

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「ResolveCase」を参照してください。

SDK for Kotlin を使用した Amazon Translate の例

次のコード例は、Amazon Translate で AWS SDK for Kotlin 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。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- シナリオ

シナリオ

Amazon SNS アプリケーションの構築

次のコード例は、サブスクリプションとパブリッシュ機能を持ち、メッセージを翻訳するアプリケーションを作成する方法を示しています。

SDK for Kotlin

Amazon SNS Kotlin API を使用して、サブスクリプションおよび発行機能を持つアプリケーションを作成する方法を示しています。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。

詳細なソースコードとウェブアプリケーションを作成する手順については、「GitHub」の詳しい例を参照してください。

詳細なソースコードと Android ネイティブアプリケーションを作成する手順については、「GitHub」の詳しい例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 SNS
- Amazon Translate

のセキュリティ AWS SDK for Kotlin

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 とお客様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。における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ます。

クラウド内のセキュリティ – お客様の責任は、使用している AWS サービス、データの機密性、組織の要件、適用可能な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 、コンプライアンスプログラムによる AWS 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

- でのデータ保護 AWS SDK for Kotlin
- AWS SDK for Kotlin TLS 1.2 のサポート
- 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

でのデータ保護 AWS SDK for Kotlin

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 Kotlin。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。ユーザーは、このインフラストラクチャでホストされ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

に関する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 AWS 責任共有モデルおよび GDPR のブログ記事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 が必須で、TLS 1.3 をお勧めします。
- 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。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- 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 などの高度な管理された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 140-3 検証済み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) 140-3」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を、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含め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 SDK for Kotlin AWS CLIまたは他の AWS のサービス を使用する場合も同様 AWS SDKs。タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

AWS SDK for Kotlin TLS 1.2 のサポート

以下の情報は、Java SSL 実装 (JVM を AWS SDK for Kotlin ターゲットとする のデフォルトの SSL 実装) にのみ適用されます。別の SSL 実装を使用している場合は、その SSL 実装を参照して、TLS バージョンを適用する方法を確認してください。

Java での TLS のサポート

TLS 1.2 は Java 7 以降でサポートされています。

TLS のバージョンを確認する方法

Java Virtual Machine (JVM) でサポートされている TLS のバージョンを確認するには、次のコードを使用します。

```
println(SSLContext.getDefault().supportedSSLParameters.protocols.joinToString(separator
 = ", "))
```

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。

```
-Djavax.net.debug=ssl
```

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 リソースの使用を許可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス 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 AWS のサービス の操作方法
- 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提供されます。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場合があります。アクセスの管理方法を理解すると、管理者に適切なアクセス許可をリクエストするのに役に立ちます。の機能にアクセスできない場合は AWS、 AWS のサービ

ス AWS ID とアクセスのトラブルシューティング「」または使用している 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 を使用する方法の詳細については AWS、使用している AWS のサービス の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 ロールを引き受けることによって、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、フェデレーティッド 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 認証情報は、フェデレーション ID の例です。フェデレーティッド ID 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、「 AWS サインイン ユーザーガイド」の「 へのサインイン AWS アカウント方法」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、 はソフトウェア開発キット (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を暗号化して署名します。AWS ツールを使用しない場合は、自分でリクエストに署名する必要があります。リクエストに自分で署名する推奨方法の使用については、「IAM ユーザーガイド」の「API リクエストに対するAWS Signature Version 4」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。たとえば、 では、アカウントのセキュリティを高めるために多要素認証 (MFA) を使用する AWS 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「多要素認証」および「IAM ユーザーガイド」の「IAM のAWS 多要素認証」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、ID プロバイダーとのフェデレーションを使用して一時的な認証情報 AWS のサービス を使用して にアクセスすることを要求します。

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、または ID ソースを介し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アイデンティティがアクセスすると AWS アカウント、ロールを引き受け、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か、独自の ID ソースのユーザーとグループのセットに接続して同期し、すべての AWS アカウント とアプリケーションで使用できます。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 ID 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、一時的な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許可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 ユーザーに関するユースケース」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 ID 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。で IAM ロールを一時的に引き受けるには AWS Management Console、ユーザーから IAM ロール (コンソール) に切り替えることができます。ロールを引き受けるには、 または AWS API オペレーションを AWS CLI 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「ロールを引き受けるための各種方法」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 – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションのロールについては、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールを作成する」を参照してください。IAM Identity Center を使用する場合は、許可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。アクセス許可セットの詳細については、「AWS IAM Identity Center User Guide」の「Permission sets」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、 (プロキシとしてロールを使用する代わりに) リソースに直接ポリシーをアタッチできます。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 でのクロスアカウントのリソースへのアクセス」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許可、サービスロール、またはサービスリンク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アクションを実行すると AWS、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 へのリクエストをリクエストする 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、削除することができます。詳細については、「IAM ユーザーガイド」の「AWS のサービスに許可を委任するロールを作成する」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、サービスリンクロールのアクセス許可を表示できますが、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。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 ロールを使用して許可を付与する」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、ID AWS またはリソースにアタッチします。ポリシーは AWS 、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 のオブジェクトです。 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) 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 JSON AWS 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のリソースに対してどのような条件下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受けることができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。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 ユーザーグループ、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 許可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ポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

アイデンティティベースのポリシーは、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、 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「管理ポリシーとインラインポリシーのいずれかを選択する」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許可を持つかを制御します。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、および Amazon VPC は AWS WAF、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、一般的でない追加のポリシータイプをサポートしています。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。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可の境界」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) の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、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ためのサービス AWS アカウント 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「サービスコントロールポリシー (SCP)」を参照してください。

- リソースコントロールポリシー (RCP) – RCP は、所有する各リソースにアタッチされた IAM ポリシーを更新することなく、アカウント内のリソースに利用可能な最大数のアクセス許可を設定するために使用できる JSON ポリシーです。RCP は、メンバーアカウントのリソースのアクセス許可を制限し、組織に属しているかどうかにかかわらず AWS アカウントのルートユーザー、 を含む ID の有効なアクセス許可に影響を与える可能性があります。RCP をサポートする のリストを含む Organizations と RCP の詳細については、AWS Organizations 「 ユーザーガイド AWS のサービス 」の「リソースコントロールポリシー (RCPs」を参照してください。 RCPs
- 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もあります。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。が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 AWS のサービス の操作方法

ほとんどの IAM 機能と AWS のサービス の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

以下の情報は、 および 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ありません AWS
- iam:PassRole を実行する権限がありません
- 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ありません AWS

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、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awes:*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:
 awes:GetWidget on resource: my-example-widget
```

この場合、awes:*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。サインイン認証情報を提供した担当者が管理者です。

iam:PassRole を実行する権限がありません

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、既存のロールをそのサービスに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。メアリーには、ロールをサービスに渡す許可がありません。

```
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 AWS のサービス の操作方法。
- 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) へのアクセスの許可」を参照してください。
- 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使用方法の違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」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AWS のサービス プログラムによる対象範囲内」を参照して、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 「 Compliance ProgramsAssurance」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、「Downloading Reports in AWS Artifact」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。 は、コンプライアンスに役立つ以下のリソース AWS を提供します。

- セキュリティのコンプライアンスとガバナンス – これらのソリューション実装ガイドでは、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の機能をデプロイする手順を示します。
- HIPAA 対応サービスのリファレンス – HIPAA 対応サービスの一覧が提供されています。すべての AWS のサービス が HIPAA の対象となるわけではありません。
- 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、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) など) にわたるセキュリティコントロールを保護および AWS のサービス 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、リソース設定が内部プラクティス、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把握できます 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一覧については、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 – 不審な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モニタリングすることで AWS アカウント、、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 を使用すると、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に対応できます。

- 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 、コンプライアンスプログラムによる AWS 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ー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、および高度に冗長なネットワークで接続された複数の物理的に分離されたアベイラビリティーゾーンを提供します。

アベイラビリティーゾーンでは、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。アベイラビリティーゾーンは、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ー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 、コンプライアンスプログラムによる AWS 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 AWS 保護されています。 AWS セキュリティサービスと がインフラスト

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。

AWS 公開された 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 が必須で、TLS 1.3 をお勧めします。
- 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 、コンプライアンスプログラムによる AWS 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。

ドキュメント履歴

このトピックでは、 AWS SDK for Kotlin デベロッパーガイドの履歴における重要な変更について説明します。

変更	説明	日付
the section called “モッキング”	SDK for Kotlin でのモックと MockK の使用に関する情報を追加する	2025 年 4 月 30 日
Gradle を使用して SDK との依存関係の競合を解決する方法に関する情報を追加する	依存関係の競合を解決するにはどうすればよいですか？	2025 年 4 月 15 日
チェックサムによるデータ整合性保護	自動チェックサム計算の詳細でコンテンツが更新されました。	2025 年 1 月 16 日
デフォルトの認証情報プロバイダーチェーンコンテンツを更新する	デフォルトの認証情報プロバイダーチェーン。	2025年 1 月 15 日
ビルドファイルの更新例	Gradle バージョン 8.11.1 で生成されたビルドファイル要素を表示します。BOM の使用を表示します。アーティファクトの最新バージョンへのリンクを埋め込みます。	2024 年 12 月 18 日
DynamoDB マッパーの追加 (開発者プレビュー) トピック	DynamoDB マッパーを使用してクラスを DynamoDB 項目にマッピングする (開発者プレビュー)	2024 年 10 月 29 日
Amazon S3 バケット名を更新する	を使用した Amazon S3 チェックサム AWS SDK for Kotlin	2024 年 9 月 30 日

OkHttp4 エンジンの情報を追加する	HTTP エンジンタイプの指定	2024 年 9 月 26 日
DynamoDB の AWS アカウントベースのエンドポイントに関する情報を追加する	AWS アカウントベースのエンドポイントを使用する	2024 年 9 月 24 日
トラブルシューティングFAQs のトピックを追加する	トラブルシューティングに関するよくある質問	2024 年 9 月 18 日
OpenTelemetry 設定の更新例とデフォルトのグローバルテレメトリプロバイダーの設定	可観測性	2024 年 5 月 2 日
サービスクライアントの作成プロセスに関する詳細を提供する	サービスクライアントを作成する	2024 年 3 月 14 日
マルチリージョンアクセスポイントの追加トピック	SDK for Kotlin を使用して Amazon S3 マルチリージョンアクセスポイントを操作する	2024 年 2 月 6 日
Gradle バージョンカタログの手順を追加する	Gradle バージョンカタログ (タブ)	2023 年 12 月 19 日
一般提供リリース	AWS SDK for Kotlin デベロッパーガイド	2023 年 11 月 27 日
SDK 更新に基づいてクライアントエンドポイント設定セクションを更新する	クライアントエンドポイント	2023 年 8 月 25 日
Amazon S3 チェックサム	Amazon S3 でフレキシブルチェックサムを使用する方法に関するセクションを追加しました。	2023 年 8 月 14 日
オブザーバビリティの追加トピック	可観測性	2023 年 8 月 3 日

再試行について説明するトピックを追加する	再試行	2023 年 7 月 7 日
SDK の更新に基づいて HTTP クライアント設定セクションを更新する	HTTP クライアント設定	2023 年 6 月 6 日
HTTP 署名前トピックを追加する	リクエストの事前署名	2023 年 6 月 2 日
HTTP インターセプターの追加トピック	HTTP インターセプター	2023 年 5 月 22 日
自動トークン更新のサポート	シングルサインオンアクセスの手順を更新します。	2023 年 5 月 18 日
Amazon S3 チェックサム	Amazon S3 でチェックサムを使用する方法を説明するセクションを追加します。	2023 年 5 月 15 日
サービスクライアント設定を上書きする	サービスクライアントの設定を上書きする方法と、リソースがどのように影響を受けるかを説明するセクションを追加します。	2023 年 5 月 8 日
最小 TLS バージョンを適用する	最小 TLS バージョンを適用するオプションを説明するセクションを追加します。	2023 年 5 月 3 日
クライアントエンドポイントの設定	クライアントエンドポイントの設定について説明するトピックを追加します。	2023 年 4 月 7 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、「IAM のセキュリティのベストプラクティス」を参照してください。	2023 年 3 月 22 日
Maven プロジェクトファイルの例を追加する	「セットアップ」トピックの「Gradle のプロジェクトファイルに加えて、Maven プロジェクトファイルの例を示します。	2022 年 12 月 2 日
デベロッパーガイドプレビューの内容を変更する	最近の開発作業を反映するように更新されたコンテンツ	2022 年 10 月 4 日
AWS SDK for Kotlin デベロッパープレビューリリース	AWS SDK for Kotlin	2021 年 12 月 2 日
AWS SDK for Kotlin アルファリリース	新しい AWS SDK for Kotlin アルファリリースの発表	2021 年 8 月 30 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。