aws

開發人員指南

適用於 Kotlin 的 AWS SDK

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

適用於 Kotlin 的 AWS SDK: 開發人員指南

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

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

什麼是 適用於 Kotlin 的 AWS SDK？ 1
開發套件入門 1
開發套件主要版本的維護與支援 1
其他資源 1
開始使用 3
步驟 1：設定本教學課程 3
步驟 2：建立專案 3
步驟 3：撰寫程式碼 6
步驟 4：建置並執行應用程式 7
成功 8
清除 8
後續步驟 8
設定 10
基本設定 10
概觀 10
AWS 存取入口網站的登入功能 11
設定單一登入 11
使用 登入 AWS CLI 12
安裝 Java 和建置工具 13
使用暫時性憑證 13
建立專案建置檔案 14
為您的專案編寫程式碼 19
使用 登入 AWS CLI 20
設定 21
建立服務用戶端 23
在程式碼中設定用戶端 24
從環境設定用戶端 24
關閉用戶端 25
AWS 區域選擇 26
預設區域供應商鏈結 26
憑證提供者 26
預設登入資料提供者鏈結 27
指定登入資料提供者 29
用戶端端點 30

自訂組態 30
範例 33
HTTP 34
HTTP 用戶端組態 34
使用 HTTP 代理 39
攔截器 39
強制執行最低 TLS 版本 41
重試 42
了解重試行為 42
自訂重試行為 46
可觀測性 51
設定 TelemetryProvider 52
指標 53
記錄 56
遙測供應商 58
覆寫用戶端組態 60
覆寫的用戶端生命週期 61
共用 資源 61
使用 SDK 63
提出請求 63
服務介面 DSL 過載 64
沒有必要輸入的請求 65
Coroutines 65
提出並行請求 65
提出封鎖請求 66
串流操作 67
串流回應 67
串流請求 68
分頁 68
等待程式 69
錯誤處理 70
服務例外狀況 70
用戶端例外狀況 71
錯誤中繼資料 71
預先簽章請求 71
預先簽章基本概念 72

進階預先簽章組態 72
預先簽署 POST 和 PUT 請求 73
軟體開發套件可以預先簽署的操作 74
疑難排解常見問答集 74
如何修正「連線已關閉」問題？ 74
為什麼在達到嘗試次數上限之前擲回例外狀況？ 75
如何修正 NoSuchMethodError或 NoClassDefFoundError？ 76
如何解決相依性衝突？ 76
模擬 78
MockK 78
使用 AWS 服務 84
Amazon S3 85
使用檢查總和保護資料完整性 86
使用多區域存取點 90
DynamoDB 95
使用以 AWS 帳戶為基礎的端點 95
使用 DynamoDB Mapper （開發人員預覽） 96
程式碼範例 120
API Gateway 121
案例 122
Aurora 122
基本概念 123
動作 135
案例 122
Auto Scaling 150
基本概念 123
動作 135
Amazon Bedrock 167
動作 135
Amazon Bedrock 執行時期 169
Amazon Nova 169
CloudWatch 173
開始使用 173
基本概念 123
動作 135
CloudWatch Logs 213

動作 135
Amazon Cognito 身分提供者 217
動作 135
案例 122
Amazon Comprehend 232
案例 122
DynamoDB 233
基本概念 123
動作 135
案例 122
Amazon EC2 261
開始使用 173
基本概念 123
動作 135
Amazon ECR 291
開始使用 173
基本概念 123
動作 135
OpenSearch Service 321
動作 135
EventBridge 325
開始使用 173
基本概念 123
動作 135
AWS Glue 356
基本概念 123
動作 135
IAM 367
基本概念 123
動作 135
AWS IoT 386
開始使用 173
基本概念 123
動作 135
AWS IoT data 410
動作 135

Amazon Keyspaces 412
開始使用 173
基本概念 123
動作 135
AWS KMS 437
動作 135
Lambda 447
基本概念 123
動作 135
案例 122
Amazon Location 456
開始使用 173
基本概念 123
動作 135
MediaConvert 488
動作 135
Amazon Pinpoint 492
動作 135
Amazon RDS 502
基本概念 123
動作 135
案例 122
Amazon RDS 資料服務 520
案例 122
Amazon Redshift 521
動作 135
案例 122
Amazon Rekognition 525
動作 135
案例 122
Route 53 網域註冊 543
開始使用 173
基本概念 123
動作 135
Amazon S3 563
基本概念 123

動作 135
案例 122
SageMaker AI 584
開始使用 173
動作 135
案例 122
Secrets Manager 609
動作 135
Amazon SES 610
案例 122
Amazon SNS 612
開始使用 173
動作 135
案例 122
Amazon SQS 639
開始使用 173
動作 135
案例 122
步驟函數 662
開始使用 173
基本概念 123
動作 135
支援 683
開始使用 173
基本概念 123
動作 135
Amazon Translate 701
案例 122
X-Ray 702
動作 135
安全 708
資料保護 708
Transport Layer Security (TLS) 709
Java 中的 TLS 支援 709
如何檢查 TLS 版本 709
身分和存取權管理 710

目標對象 710
使用身分驗證 710
使用政策管理存取權 712
AWS 服務如何使用 IAM 713
對AWS身分和存取進行故障診斷 713
合規驗證 715
恢復能力 715
基礎設施安全性 715
文件歷史紀錄 717
..... dccxx

什麼是 適用於 Kotlin 的 AWS SDK？

為 Amazon Web Services 適用於 Kotlin 的 AWS SDK 提供 Kotlin APIs。使用 SDK，您可以建置可搭配 Amazon S3、Amazon EC2、Amazon DynamoDB 等使用的 Kotlin 應用程式。使用 Kotlin 開發套件，您可以鎖定 JVM 平台或 Android API 第 24 級或更高層級。未來版本將支援 JavaScript 和 Native 等其他平台。

若要追蹤未來版本中即將推出的功能，請參閱 GitHub 上的藍圖。

開發套件入門

若要開始使用 SDK，請遵循開始使用教學課程。

若要設定開發環境，請參閱 設定。

若要建立和設定服務用戶端以向 提出請求 AWS 服務，請參閱組態。如需 SDK 各種功能的資訊，請參閱 使用 SDK。

如需執行特定 API 操作的使用案例和範例，請參閱 程式碼範例。

開發套件主要版本的維護與支援

如需 SDK 主要版本及其基礎相依性維護和支援的相關資訊，請參閱 AWS SDKs和工具參考指南中的下列主題：

- AWS SDKs和工具維護政策
- AWS SDKs和工具版本支援矩陣

其他資源

除了本指南之外，以下是適用於 Kotlin 開發人員的 SDK 的寶貴線上資源：

- AWS 開發人員部落格
- 開發人員論壇
- SDK 來源 (GitHub)
- AWS 程式碼範例目錄

- @awsdevelopers (X，先前為 Twitter)

開始使用適用於 Kotlin 的 SDK

為每個適用於 Kotlin 的 AWS SDK提供 Kotlin APIs AWS 服務。您可以使用 SDK 建置可搭配 Amazon S3、Amazon EC2、Amazon DynamoDB 等使用的 Kotlin 應用程式。

本教學課程說明如何使用 Gradle 來定義 的相依性適用於 Kotlin 的 AWS SDK。然後，您可以建立程式碼，將資料寫入 DynamoDB 資料表。雖然您可能想要使用 IDE 的功能，但本教學課程只需要終端機視窗和文字編輯器。

請依照下列步驟完成本教學課程：

- 步驟 1：設定本教學課程
- 步驟 2：建立專案
- 步驟 3：撰寫程式碼
- 步驟 4：建置並執行應用程式

步驟 1：設定本教學課程

開始本教學課程之前，您需要可以存取 DynamoDB 的 IAM Identity Center 許可集，而且需要以 IAM Identity Center 單一登入設定設定的 Kotlin 開發環境才能存取AWS。

請依照基本設定本指南中的指示，取得本教學課程的基本設定。

使用 Kotlin SDK 的單一登入存取設定開發環境，並擁有作用中的AWS存取入口網站工作階段後，請繼續步驟 2。

步驟 2：建立專案

若要建立本教學課程的專案，請先使用 Gradle 為 Kotlin 專案建立基本檔案。然後，使用 所需的設定、相依性和程式碼來更新檔案適用於 Kotlin 的 AWS SDK。

使用 Gradle 建立新專案

> **Note**
>
> 本教學課程使用 Gradle 8.11.1 版搭配 `gradle init`命令，在下面的步驟 3 中提供五個提示。如果您使用不同的 Gradle 版本，提示和預先填入的成品版本可能不同。

1. 在您選擇的getstarted位置建立新的目錄，例如桌面或主資料夾。
2. 開啟終端機或命令提示視窗，然後導覽至您建立的getstarted目錄。
3. 使用以下命令建立新的 Gradle 專案和基本 Kotlin 類別。

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

   - 當系統提示輸入目標 時Java version，按 Enter（預設為 21)。
   - 出現 提示時Project name，按 Enter（在本教學getstarted課程中預設為目錄名稱）。
   - 出現提示時application structure，按 Enter（預設為 Single application project)。
   - 出現 提示時Select test framework，按 Enter（預設為 kotlin.test)。
   - 出現 提示時Generate build using new APIs and behavior，按 Enter（預設為 no)。

使用 和適用於 Kotlin 的 AWS SDK Amazon S3 的相依性來設定專案

- 在您先前程序中建立的getstarted目錄中，將settings.gradle.kts檔案的內容取代為下列內容，將 *X.Y.Z* 取代為最新版本https://github.com/awslabs/aws-sdk-kotlin/releases/latest的適用於 Kotlin 的 SDK：

```
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")
```

- 導覽至 gradle目錄內的 getstarted目錄。將名為 的版本目錄檔案的內容取代libs.versions.toml為下列內容：

```
[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" }
```

- 導覽至 app 目錄並開啟 build.gradle.kts 檔案。將其內容替換為下列程式碼，然後儲存您的變更：

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


dependencies {
    implementation(awssdk.services.s3) // Add dependency on the ### Kotlin # AWS
 SDK'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 區段包含 Amazon S3 模組的 implementation項目適用於 Kotlin 的 AWS SDK。Gradle 編譯器在 java區段中設定為使用 Java 21。

步驟 3：撰寫程式碼

建立並設定專案之後，請編輯專案的預設類別App，以使用下列範例程式碼。

1. 在您的專案資料夾中app，導覽至目錄 src/main/kotlin/org/example。開啟 App.kt 檔案。
2. 以下列程式碼取代其內容並儲存檔案。

```
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!")
}
```

步驟 4：建置並執行應用程式

建立專案並包含範例類別之後，請建置並執行應用程式。

1. 開啟終端機或命令提示視窗，然後導覽至您的專案目錄 getstarted。
2. 使用下列命令來建置和執行您的應用程式：

```
gradle run
```

> **Note**
>
> 如果您取得 `IdentityProviderException`，您可能沒有作用中的單一登入工作階段。執行 `aws sso login`AWS CLI 命令來啟動新的工作階段。

應用程式會呼叫 createBucket API 操作來建立新的 S3 儲存貯體，然後呼叫 putObject 將新物件放入新的 S3 儲存貯體。

在結束時的 `cleanUp()`函數中，應用程式會刪除物件，然後刪除 S3 儲存貯體。

在 Amazon S3 主控台中查看結果

1. 在 中`App.kt`，註解`runBlocking`區段`cleanUp(s3)`中的行，並儲存檔案。
2. 執行 來重建專案，並將新物件放入新的 S3 儲存貯體`gradle run`。
3. 登入 Amazon S3 主控台，以檢視新 S3 儲存貯體中的新物件。

檢視物件之後，請刪除 S3 儲存貯體。

成功

如果您的 Gradle 專案建置並執行時沒有錯誤，恭喜您。您已成功使用 建置第一個 Kotlin 應用程式適用於 Kotlin 的 AWS SDK。

清除

當您完成開發新的應用程式時，請刪除您在本教學課程中建立的任何AWS資源，以避免產生任何費用。您也可以刪除或封存您在步驟 2 中建立的專案資料夾 (`get-started`)。

請依照下列步驟清除資源：

- 如果您已對 `cleanUp()`函數的呼叫進行註解，請使用 Amazon S3 主控台刪除 S3 儲存貯體。Amazon S3

後續步驟

現在您已完成基本概念，您可以了解以下內容：

- 使用適用於 Kotlin 的 SDK 的其他設定步驟

- 適用於 Kotlin 的 SDK 組態
- 使用適用於 Kotlin 的 SDK
- 適用於 Kotlin 的 SDK 安全性

設定 適用於 Kotlin 的 AWS SDK

若要 AWS 服務 使用 向 提出請求 適用於 Kotlin 的 AWS SDK，您需要下列項目：

- 登入 AWS 存取入口網站的能力
- 使用應用程式所需 AWS 資源的許可
- 具有下列元素的開發環境：
  - 使用下列至少一種方式設定的共用組態檔案：
    - `config` 檔案包含 IAM Identity Center 登入資料設定，讓 SDK 可以取得 AWS 登入資料
    - `credentials` 檔案包含臨時登入資料
  - 建置自動化工具，例如 Gradle 或 Maven
- 當您準備好執行應用程式時，作用中的 AWS 存取入口網站工作階段

在本主題中

- 基本設定
- 建立專案建置檔案
- 使用適用於 Kotlin 的 SDK 為您的 Kotlin 專案編寫程式碼

基本設定

概觀

若要成功開發 AWS 服務 使用 存取的應用程式 適用於 Kotlin 的 AWS SDK，必須符合下列要求。

- 您必須能夠登入 AWS 中提供的存取入口網站 AWS IAM Identity Center。
- 為 SDK 設定的 IAM 角色許可必須允許存取 AWS 服務 您的應用程式所需的 。與 PowerUserAccess AWS 受管政策相關聯的許可足以滿足大多數開發需求。
- 具有下列元素的開發環境：
  - 至少以下列其中一種方式設定的共用組態檔案：
    - `config` 檔案包含 IAM Identity Center 單一登入設定，讓 SDK 可以取得 AWS 登入資料。
    - `credentials` 檔案包含臨時登入資料。
  - 安裝 Java 8 或更新版本。

- Maven 或 Gradle 等建置自動化工具。
- 使用程式碼的文字編輯器。
- （選用，但建議使用）IDE（整合式開發環境），例如 IntelliJ IDEA 或 Eclipse。

  使用 IDE 時，您也可以整合 AWS 工具組以更輕鬆地使用 AWS 服務。AWS Toolkit for IntelliJ 和 AWS Toolkit for Eclipse是您可以使用的兩個工具組。

- 當您準備好執行應用程式時，作用中的 AWS 存取入口網站工作階段。您可以使用 AWS Command Line Interface 啟動 IAM Identity Center 存取入口網站的登入程序。 AWS

> **Important**
>
> 此設定區段中的指示假設您或組織使用 IAM Identity Center。如果您的組織使用獨立於 IAM Identity Center 運作的外部身分提供者，請了解如何取得適用於 Kotlin 的 SDK 暫時登入資料。請依照這些指示，將臨時登入資料新增至 `~/.aws/credentials` 檔案。
> 如果您的身分提供者自動將臨時登入資料新增至 `~/.aws/credentials` 檔案，請確定設定檔名稱為 ，`[default]`如此您就不需要提供設定檔名稱給 SDK 或 AWS CLI。

AWS 存取入口網站的登入功能

AWS 存取入口網站是您手動登入 IAM Identity Center 的 Web 位置。URL 的格式為 `d-xxxxxxxxxx.awsapps.com/start`或 *`your_subdomain`*`.awsapps.com/start`。

如果您不熟悉 AWS 存取入口網站，請遵循《 AWS SDKs和工具參考指南》中的 IAM Identity Center 身分驗證主題中的帳戶存取指引。

設定 SDK 的單一登入存取

完成程式設計存取區段中的步驟 2 後，軟體開發套件才能使用 IAM Identity Center 身分驗證，您的系統應該包含下列元素。

- 在執行應用程式之前 AWS CLI，您用來啟動AWS 存取入口網站工作階段的 。
- 包含預設設定檔`~/.aws/config`的檔案。適用於 Kotlin 的 SDK 使用描述檔的 SSO 字符提供者組態，在傳送請求至 之前取得憑證 AWS。該 `sso_role_name` 值是連接到 IAM Identity Center 許可集合的 IAM 角色，應該允許存取應用程式中使用的 AWS 服務 。

下列範例config檔案顯示使用 SSO 權杖提供者組態設定的預設設定檔。設定檔的 sso_session 設定是指已命名的 sso-session 區段。sso-session 區段包含啟動 AWS 存取入口網站工作階段的設定。

```
[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
```

如需 SSO 權杖提供者組態中使用的設定詳細資訊，請參閱 AWS SDKs和工具參考指南中的 SSO 權杖提供者組態。

如果您的開發環境未如先前所示設定程式設計存取，請遵循 SDKs參考指南中的步驟 2。

使用 登入 AWS CLI

在執行存取的應用程式之前 AWS 服務，您需要作用中 AWS 的存取入口網站工作階段，開發套件才能使用 IAM Identity Center 身分驗證來解析登入資料。在 中執行下列命令 AWS CLI ，以登入 AWS 存取入口網站。

```
aws sso login
```

由於您有預設的設定檔設定，您不需要使用 --profile選項呼叫 命令。如果您的 SSO 權杖提供者組態使用具名設定檔，則命令為 aws sso login --profile *named-profile*。

若要測試您是否已有作用中的工作階段，請執行下列 AWS CLI 命令。

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

對此命令的回應，應報告共用 config 檔案中設定的 IAM Identity Center 帳戶和許可集合。

> **Note**
>
> 如果您已經擁有作用中的 AWS 存取入口網站工作階段並執行 `aws sso login`，則不需要提供憑證。
> 不過，您會看到一個對話方塊，要求 `botocore` 存取您資訊的許可。 `botocore`是 的基礎 AWS CLI 。
> 選取允許以授權存取適用於 Kotlin 的 AWS CLI 和 SDK 的資訊。

安裝 Java 和建置工具

您的開發環境需要下列項目：

- JDK 8 或更新版本。 適用於 Kotlin 的 AWS SDK 適用於 Oracle Java SE 開發套件和 Open Java 開發套件 (OpenJDK) 的發行版本，例如 Amazon Corretto、Red Hat OpenJDK 和 AdoptOpenJDK。
- 支援 Maven Central 的建置工具或 IDE，例如 Apache Maven、Gradle 或 IntelliJ。
  - 如需如何安裝和使用 Maven 的詳細資訊，請參閱 https：//http://maven.apache.org/。
  - 如需如何安裝和使用 Gradle 的詳細資訊，請參閱 https：//https://gradle.org/。
  - 如需如何安裝和使用 IntelliJ IDEA 的資訊，請參閱 https：//https://www.jetbrains.com/idea/。

使用暫時性憑證

除了設定 SDK 的 IAM Identity Center 單一登入存取之外，您也可以使用暫時登入資料來設定開發環境。

為臨時登入資料設定本機登入資料檔案

1. 建立共用登入資料檔案
2. 在登入資料檔案中，貼上下列預留位置文字，直到您貼上運作中的臨時登入資料：

```
[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. 儲存檔案。檔案現在`~/.aws/credentials`應該存在於您的本機開發系統上。如果未指定特定具名設定檔，此檔案包含適用於 Kotlin 的 SDK 所使用的 【預設】 設定檔。

4. 登入 AWS 存取入口網站
5. 請遵循手動登入資料重新整理標題下的這些指示，從 AWS 存取入口網站複製 IAM 角色登入資料。

   a. 針對連結指示中的步驟 4，選擇 IAM 角色名稱，為您的開發需求授予存取權。此角色通常具有 PowerUserAccess 或 Developer 等名稱。

   b. 針對步驟 7，選取手動將設定檔新增至您的 AWS 登入資料檔案選項，然後複製內容。
6. 將複製的登入資料貼到本機`credentials`檔案中，並移除產生的設定檔名稱。您的 檔案應該類似以下內容：

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

7. 儲存`credentials`檔案

適用於 Kotlin 的 SDK 會在建立服務用戶端時存取這些臨時登入資料，並用於每個請求。在步驟 5a 中選擇的 IAM 角色設定會決定臨時憑證的有效時長。最長持續時間為 12 小時。

臨時憑證過期後，請重複步驟 4 至 7。

建立專案建置檔案

在您設定單一登入存取和開發環境之後，請使用您偏好的建置工具建立 Kotlin 專案。在建置檔案中，指定應用程式需要存取 AWS 服務 之 的相依性。

若要查看所有可能的 Maven 成品名稱清單，請參閱 API 參考文件。若要尋找最新版本的 SDK，請檢查 GitHub 上的最新版本。

下列範例建置檔案提供使用 7 開始編碼專案的必要元素 AWS 服務。

Gradle

適用於 Kotlin 的 AWS SDK 發佈 Gradle 版本目錄和物料清單 (BOM)，可協助您探索相依性的名稱，並保持跨多個成品的版本編號同步。

請注意，版本目錄是 Gradle 第 8 版之前的預覽功能。根據您使用的 Gradle 版本，您可能需要透過特徵預覽 API 選擇加入。

使用 Gradle 版本目錄

1. 在您的 settings.gradle.kts檔案中，在 versionCatalogs區塊內新增 dependencyResolutionManagement區塊。

   下列範例檔案會設定 適用於 Kotlin 的 AWS SDK 版本目錄。您可以導覽至 *X.Y.Z* 連結，以查看可用的最新版本。

```
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. build.gradle.kts 使用版本目錄提供的類型安全識別符，宣告 中的相依性。

   下列範例檔案會宣告七個 的相依性 AWS 服務。

```
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 版本，例如 17或 21。

Maven

下列範例pom.xml檔案具有七個 的相依性 AWS 服務。您可以導覽至 *X.Y.Z* 連結，以查看可用的最新版本。

```
<?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 版本，例如 17或 21。

使用適用於 Kotlin 的 SDK 為您的 Kotlin 專案編寫程式碼

現在，樂趣開始了。當您開發應用程式時，您可以參考 適用於 Kotlin 的 AWS SDK API 參考，以取得 API 操作的完整資訊。使用以下連結取得一般 Kotlin API 資訊：

- 標準程式庫 API 參考
- Coroutines 概觀
- Coroutines API

使用 登入 AWS CLI

每當您執行存取的程式時 AWS 服務，都需要作用中 AWS 的存取入口網站工作階段。您執行下列 命令，即可進行新增：

```
aws sso login
```

由於您有預設的設定檔設定，因此您不需要使用 `--profile` 選項呼叫指令。如果您的 IAM Identity Center 單一登入組態使用具名設定檔，則命令為 `aws sso login --profile named-profile`。

若要測試您是否已經有作用中的工作階段，請執行下列 AWS CLI 命令。

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

對此命令的回應，應報告共用 `config` 檔案中設定的 IAM Identity Center 帳戶和許可集合。

設定適用於 Kotlin 的 AWS SDK

本節說明如何使用 設定服務用戶端適用於 Kotlin 的 AWS SDK。如需詳細資訊，請參閱 SDK 和工具參考指南，其中包含適用於AWS SDKs的組態概觀。

內容

- 建立服務用戶端
  - 在程式碼中設定用戶端
  - 從環境設定用戶端
  - 關閉用戶端
- AWS 區域選擇
  - 預設區域供應商鏈結
- 憑證提供者
  - 預設登入資料提供者鏈結
    - 了解預設登入資料提供者鏈結
  - 指定登入資料提供者
    - 使用獨立提供者快取登入資料
- 設定用戶端端點
  - 自訂組態
    - 設定 endpointUrl
    - 設定 endpointProvider
      - EndpointProvider 屬性
    - endpointUrl 或 endpointProvider
    - 有關 Amazon S3 的備註
  - 範例
    - endpointUrl 範例
    - endpointProvider 範例
    - endpointUrl 和 endpointProvider
- HTTP
  - HTTP 用戶端組態
    - 基本組態

- 匯入
- Code
- 進階組態
  - 指定 HTTP 引擎類型
    - 匯入
    - Code
    - 使用 OkHttp4Engine
    - 使用明確的 HTTP 用戶端
      - 匯入
      - Code
  - 閒置連線監控
    - 匯入
    - Code
- 使用 HTTP 代理
  - 使用 JVM 系統屬性
  - 使用環境變數
  - 在 EC2 執行個體上使用代理
- HTTP 攔截器
  - 攔截器註冊
    - 所有服務用戶端操作的攔截器
    - 僅特定操作的攔截器
- 強制執行最低 TLS 版本
  - 設定 HTTP 引擎
  - 設定 sdk.minTls JVM 系統屬性
  - 設定SDK_MIN_TLS環境變數
- 中的重試次數適用於 Kotlin 的 AWS SDK
- 了解重試行為
  - 預設重試組態
  - 哪些例外狀況可以重試？
    - 依錯誤碼重試

        - 可透過 HTTP 狀態碼重試
        - 依錯誤類型重試
        - 可由 SDK 中繼資料重試
    - 檢查例外狀況是否可重試
    - 當重試失敗時，哪些例外狀況會到達您的程式碼
  - 自訂重試行為
    - 設定最大嘗試次數
    - 設定延遲和退避
    - 設定重試權杖儲存貯體
    - 設定自適應重試
- 可觀測性
  - 設定 TelemetryProvider
    - 設定預設全域遙測供應商
    - 為特定服務用戶端設定遙測供應商
  - 指標
  - 記錄
    - 指定線路層級訊息的日誌模式
      - 在程式碼中設定日誌模式
      - 從環境設定日誌模式
  - 遙測供應商
    - 設定 OpenTelemetry 型遙測供應商
      - 先決條件
      - 設定軟體開發套件
      - Resources
- 覆寫服務用戶端組態
  - 覆寫用戶端的生命週期
  - 用戶端之間共用的資源

建立服務用戶端

若要向 提出請求AWS 服務，您必須先執行個體化該服務的用戶端。

您可以設定服務用戶端的常見設定，例如要使用的 HTTP 用戶端、記錄層級和重試組態。此外，每個服務用戶端都需要AWS 區域和登入資料提供者。SDK 使用這些值將請求傳送至正確的區域，並使用正確的登入資料簽署請求。

您可以在程式碼中以程式設計方式指定這些值，或從環境中自動載入這些值。

在程式碼中設定用戶端

若要設定具有特定值的服務用戶端，您可以在傳遞至服務用戶端原廠方法的 lambda 函數中指定它們，如下列程式碼片段所示。

```
val dynamoDbClient = DynamoDbClient {
    region = "us-east-1"
    credentialsProvider = ProfileCredentialsProvider(profileName = "myprofile")
}
```

您在組態區塊中未指定的任何值都會設定為預設值。例如，如果您未指定先前程式碼的登入資料提供者，登入資料提供者會預設為預設的登入資料提供者鏈結。

> **Warning**
>
> 有些屬性 例如 `region` 沒有預設值。使用程式設計組態時，您必須在組態區塊中明確指定它們。如果 SDK 無法解析 屬性，API 請求可能會失敗。

從環境設定用戶端

建立服務用戶端時，軟體開發套件可以檢查目前執行環境中的位置，以判斷一些組態屬性。這些位置包括共用組態和登入資料檔案、環境變數和 JVM 系統屬性。可解析的屬性包括AWS區域、重試策略、日誌模式和其他。如需軟體開發套件可從執行環境解析之所有設定的詳細資訊，請參閱軟體AWS SDKs與工具設定參考指南。

若要建立具有環境來源組態的用戶端，請在服務用戶端界面`suspend fun fromEnvironment()`上使用靜態方法：

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

以這種方式建立用戶端在 Amazon EC2 上執行時很有用AWS Lambda，或可在環境中使用服務用戶端組態的任何其他內容。這會將您的程式碼與執行環境分離，讓您更輕鬆地將應用程式部署到多個區域，而無需變更程式碼。

此外，您可以透過將 lambda 區塊傳遞至 來覆寫特定屬性`fromEnvironment`。下列範例會從環境（例如區域）載入一些組態屬性，但會特別覆寫登入資料提供者，以使用來自設定檔的登入資料。

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

開發套件會針對無法從程式設計設定或環境判斷的任何組態屬性，使用預設值。例如，如果您未在程式碼或環境設定中指定登入資料提供者，登入資料提供者會預設為預設的登入資料提供者鏈結。

> **Warning**
>
> 區域之類的某些屬性沒有預設值。您必須在環境設定中或在組態區塊中明確指定它們。如果 SDK 無法解析 屬性，API 請求可能會失敗。

> **Note**
>
> 雖然在執行環境中可以找到與登入資料相關的屬性，例如臨時存取金鑰和 SSO 組態，但用戶端在建立時不會取得這些值。反之，登入資料提供者層會在每次請求時存取這些值。

關閉用戶端

當您不再需要服務用戶端時，請將其關閉以釋出其使用的任何資源：

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

由於服務用戶端擴展了`Closeable`界面，因此您可以使用 `use`擴充功能，在區塊結尾自動關閉用戶端，如下列程式碼片段所示。

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

在先前的範例中，lambda 區塊會收到剛建立之用戶端的參考。您可以在此用戶端參考和區塊完成時叫用操作，包括擲回例外狀況時，用戶端會關閉。

AWS 區域選擇

使用AWS 區域，您可以存取在特定地理區域中操作AWS 服務的 。這對於備援以及讓您的資料和應用程式在靠近您和您的使用者存取位置附近執行，都很有用。

預設區域供應商鏈結

從環境載入服務用戶端的組態時，會使用下列查詢程序：

1. 建置器上設定的任何明確區域。
2. 已檢查 `aws.region` JVM 系統屬性。如果已設定，則該區域會用於用戶端的組態。
3. 檢查 `AWS_REGION` 環境變數。如果已設定，則該區域會用於用戶端的組態。
   a. 注意：此環境變數是由 Lambda 容器設定。
4. SDK 會檢查AWS共用的組態檔案。如果為作用中設定檔設定 `region` 屬性，則 SDK 會使用它。
   a. `AWS_CONFIG_FILE` 環境變數可用於自訂共用組態檔的位置。
   b. `aws.profile` JVM 系統屬性或`AWS_PROFILE`環境變數可用來自訂 SDK 載入的設定檔。
5. SDK 會嘗試使用 Amazon EC2 執行個體中繼資料服務來判斷目前執行中 EC2 執行個體的區域。
6. 如果區域目前仍未解決，則用戶端建立會失敗，但出現例外狀況。

憑證提供者

> ⚠ 預設登入資料提供者鏈結解析 1.4.0 版所變更登入資料的順序。如需詳細資訊，請參閱以下備註。

當您使用 將請求傳送至 Amazon Web Services 時適用於 Kotlin 的 AWS SDK，必須使用 發行的登入資料以密碼編譯方式簽署請求AWS。Kotlin SDK 會自動為您簽署請求。若要取得登入資料，軟體開發套件可以使用位於多個位置的組態設定，例如 JVM 系統屬性、環境變數、共用AWS`config`和`credentials`檔案，以及 Amazon EC2 執行個體中繼資料。

SDK 使用登入資料提供者抽象來簡化從各種來源擷取登入資料的程序。開發套件包含數個登入資料提供者實作。

例如，如果擷取的組態包含來自共用`config`檔案的 IAM Identity Center 單一登入存取設定，則 SDK 會與 IAM Identity Center 搭配使用，以擷取用於提出請求的臨時憑證AWS 服務。透過這種取得憑證

的方法，開發套件會使用 IAM Identity Center 提供者（也稱為 SSO 憑證提供者）。本指南的設定區段說明了此組態。

若要使用特定的登入資料提供者，您可以在建立服務用戶端時指定一個登入資料提供者。或者，您可以使用預設登入資料提供者鏈結來自動搜尋組態設定。

預設登入資料提供者鏈結

在用戶端建構時未明確指定時，適用於 Kotlin 的 SDK 會使用登入資料提供者，依序檢查您可以提供登入資料的每個位置。此預設登入資料提供者會實作為登入資料提供者鏈。

若要使用預設鏈結在應用程式中提供登入資料，請建立 服務用戶端，而不明確提供 `credentialsProvider` 屬性。

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

如需建立服務用戶端的詳細資訊，請參閱建構和設定用戶端。

了解預設登入資料提供者鏈結

預設登入資料提供者鏈結會使用下列預先定義的序列搜尋登入資料組態。當設定的設定提供有效的登入資料時，鏈結會停止。

1. AWS存取金鑰 (JVM 系統屬性）

   SDK 會尋找 `aws.accessKeyId`、 `aws.secretAccessKey`和 `aws.sessionToken` JVM 系統屬性。

2. AWS存取金鑰（環境變數）

   SDK 會嘗試從 `AWS_ACCESS_KEY_ID`和 `AWS_SECRET_ACCESS_KEY`以及 `AWS_SESSION_TOKEN`環境變數載入登入資料。

3. Web 身分字符

   SDK 會尋找環境變數 `AWS_WEB_IDENTITY_TOKEN_FILE`和 `AWS_ROLE_ARN`（或 JVM 系統屬性 `aws.webIdentityTokenFile`和 `aws.roleArn`)。根據字符資訊和角色，開發套件會取得臨時登入資料。

4. 組態檔案中的設定檔

   在此步驟中，軟體開發套件會使用與設定檔相關聯的設定。根據預設，開發套件會使用共用AWSconfig的 和 credentials 檔案，但如果已設定AWS_CONFIG_FILE環境變數，開發套件會使用該值。如果未設定AWS_PROFILE環境變數（或 aws.profile JVM 系統屬性），軟體開發套件會尋找「預設」設定檔，否則會尋找符合AWS_PROFILE's值的設定檔。

   SDK 會根據前一段所述的組態尋找設定檔，並使用其中定義的設定。如果 SDK 找到的設定包含不同登入資料提供者方法的設定組合，則 SDK 會使用下列順序：

   a. AWS存取金鑰（組態檔案） - SDK 使用 aws_access_key_id、 aws_access_key_id和 的設定aws_session_token。

   b. 擔任角色組態 - 如果 SDK 找到 role_arn和 source_profile 或 credential_source設定，它會嘗試擔任角色。如果 SDK 找到 source_profile設定，它會從另一個設定檔中取得登入資料，以接收 所指定角色的臨時登入資料role_arn。如果 SDK 找到credential_source設定，它會根據credential_source設定的值，從 Amazon ECS 容器、Amazon EC2 執行個體或環境變數中取得登入資料。然後，它會使用這些登入資料來取得角色的臨時登入資料。

      設定檔應包含 source_profile設定或 credential_source設定，但不能同時包含兩者。

   c. Web 身分字符組態 - 如果 SDK 找到 role_arn 和 web_identity_token_file設定，它會取得臨時登入資料，以根據 role_arn和字符存取AWS資源。

   d. SSO 字符組態 - 如果軟體開發套件找到 sso_session、sso_account_id、sso_role_name 設定（以及組態檔案中的配套sso-session區段），軟體開發套件會從 IAM Identity Center 服務擷取臨時憑證。

   e. 舊版 SSO 組態 - 如果 SDK 找到 sso_start_url、sso_account_id、 sso_region和sso_role_name設定，則 SDK 會從 IAM Identity Center 服務擷取臨時憑證。

   f. 登入組態 - 如果 SDK 找到login_session設定，它會使用登入工作階段的臨時登入資料，或在 5 分鐘內過期時嘗試重新整理這些登入資料。若要了解如何啟動登入工作階段，請參閱 AWS CLI 使用者指南。

   g. 程序組態 - 如果 SDK 找到credential_process設定，則會使用路徑值來叫用程序並取得臨時登入資料。

5. 容器憑證

   SDK 會尋找環境變數AWS_CONTAINER_CREDENTIALS_RELATIVE_URI或
   AWS_CONTAINER_CREDENTIALS_FULL_URI和
   AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE或

AWS_CONTAINER_AUTHORIZATION_TOKEN。它使用這些值透過 GET 請求從指定的 HTTP 端點載入登入資料。

6. IMDS 登入資料

   軟體開發套件會嘗試在預設或設定的 HTTP 端點上，從執行個體中繼資料服務擷取登入資料。SDK 僅支援 IMDSv2。

如果目前仍無法解析登入資料，則用戶端建立會失敗，但有例外狀況。

> **注意：依登入資料解析順序進行變更**
>
> 上述的登入資料解析順序是1.4.x+最新版本的適用於 Kotlin 的 SDK。在1.4.0發行之前，項目編號 3 和 4 已切換，而目前的 4a 項目遵循目前的 4g 項目。

指定登入資料提供者

您可以指定登入資料提供者，而不是使用預設的提供者鏈結。此方法可讓您直接控制 SDK 使用哪些登入資料。

例如，若要使用擔任 IAM 角色的登入資料，請在建立用戶端StsAssumeRoleCredentialsProvider時指定 ：

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

您也可以建立自訂鏈結 (CredentialsProviderChain)，以您偏好的順序結合多個供應商。

使用獨立提供者快取登入資料

> **Important**
>
> 預設鏈結會自動快取登入資料。獨立提供者不會快取登入資料。若要避免在每次 API 呼叫時擷取登入資料，請使用 包裝您的提供者CachedCredentialsProvider。快取的供應商只會在目前的登入資料過期時擷取新的登入資料。

若要使用獨立提供者快取登入資料，請使用 CachedCredentialsProvider類別：

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

或者，使用 cached() 延伸函數以取得更簡潔的程式碼：

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

設定用戶端端點

當適用於 Kotlin 的 AWS SDK呼叫 時AWS 服務，第一個步驟是判斷路由請求的位置。此程序稱為端點解析。

您可以在建置服務用戶端時設定 SDK 的端點解析。端點解析的預設組態通常沒問題，但有幾個原因可能會導致您修改預設組態。兩個範例原因如下：

- 向服務的發行前版本或服務的本機部署提出請求。
- 存取 SDK 中尚未建模的特定服務功能。

> **Warning**
>
> 端點解析是進階 SDK 主題。如果您變更預設設定，則會有損壞程式碼的風險。預設設定應適用於生產環境中的大多數使用者。

自訂組態

您可以使用建置用戶端時可用的兩個屬性來自訂服務用戶端的端點解析：

1. endpointUrl: Url
2. endpointProvider: EndpointProvider

設定 **endpointUrl**

您可以為 設定值endpointUrl，以指出服務的「基礎」主機名稱。不過，此值不是最終值，因為它會以參數形式傳遞給用戶端的EndpointProvider執行個體。EndpointProvider 實作接著可以檢查並可能修改該值，以判斷最終端點。

例如，如果您指定 Amazon Simple Storage Service (Amazon S3) 用戶端endpointUrl的值並執行GetObject操作，預設端點提供者實作會將儲存貯體名稱注入主機名稱值。

實際上，使用者將 endpointUrl值設定為指向服務的開發或預覽執行個體。

設定 **endpointProvider**

服務用戶端的EndpointProvider實作決定最終端點解析。下列程式碼區塊中顯示的EndpointProvider界面公開了 resolveEndpoint方法。

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

服務用戶端會針對每個請求呼叫 resolveEndpoint方法。服務用戶端使用提供者傳回Endpoint的值，不會進一步變更。

EndpointProvider 屬性

resolveEndpoint 方法接受服務特定的EndpointParameters物件，其中包含端點解析中使用的屬性。

每個服務都包含下列基本屬性。

名稱	類型	Description
region	String	用戶端AWS的區域
endpoint	String	的值集的字串表示法 endpointUrl
useFips	Boolean	是否在用戶端的組態中啟用 FIPS 端點

名稱	類型	Description
useDualStack	Boolean	是否在用戶端的組態中啟用雙堆疊端點

服務可以指定解析所需的其他屬性。例如，Amazon S3 S3EndpointParameters包含儲存貯體名稱和數個 Amazon S3-specific功能設定。例如， forcePathStyle 屬性會判斷是否可以使用虛擬主機定址。

如果您實作自己的提供者，則不需要建構自己的 執行個體EndpointParameters。開發套件提供每個請求的屬性，並將其傳遞給您的 實作resolveEndpoint。

endpointUrl 或 endpointProvider

請務必了解，下列兩個陳述式不會產生具有同等端點解析行為的用戶端：

```
// 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")
    }
}
```

設定 endpointUrl 屬性的 陳述式會指定傳遞給 （預設） 供應商的基本 URL，該 URL 可以修改為端點解析的一部分。

設定 的 陳述式會endpointProvider指定 S3Client使用的最終 URL。

雖然您可以設定這兩個屬性，但在大多數需要自訂的情況下，您可以提供其中一個屬性。身為一般 SDK 使用者，您最常提供 endpointUrl值。

有關 Amazon S3 的備註

Amazon S3 是一種複雜的服務，其許多功能透過自訂端點自訂建模，例如儲存貯體虛擬託管。虛擬託管是 Amazon S3 的一項功能，其中儲存貯體名稱會插入主機名稱。

因此，建議您不要取代 Amazon S3 服務用戶端中的EndpointProvider實作。如果您需要擴展其解決行為，可能透過傳送請求到具有其他端點考量的本機開發堆疊，我們建議您包裝預設實作。下列endpointProvider範例顯示此方法的範例實作。

範例

endpointUrl 範例

下列程式碼片段顯示如何覆寫 Amazon S3 用戶端的一般服務端點。

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

endpointProvider 範例

下列程式碼片段說明如何提供自訂端點提供者，以包裝 Amazon S3 的預設實作。

```
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 和 endpointProvider

下列範例程式示範 endpointUrl和 endpointProvider設定之間的互動。這是進階使用案例。

```
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

本節涵蓋 中 HTTP 相關設定的組態適用於 Kotlin 的 AWS SDK。

主題

- HTTP 用戶端組態
- 使用 HTTP 代理
- HTTP 攔截器
- 強制執行最低 TLS 版本

HTTP 用戶端組態

根據預設，適用於 Kotlin 的 AWS SDK會使用以 OkHttp 為基礎的 HTTP 用戶端。您可以透過提供明確設定的用戶端來覆寫 HTTP 用戶端及其組態。

> **Warning**
>
> 無論您使用哪種 HTTP 引擎，專案中的其他相依性都可能有與 SDK 所需的特定引擎版本衝突的暫時性相依性。特別是 Spring Boot 等架構已知會管理 OkHttp 等相依性，並依賴比 SDK 更舊的版本。如需詳細資訊the section called “如何解決相依性衝突？”，請參閱 。

> **Note**
>
> 根據預設，每個服務用戶端都會使用自己的 HTTP 用戶端複本。如果您在應用程式中使用多個服務，建議您建構單一 HTTP 用戶端，並將其分享給所有服務用戶端。

基本組態

當您設定服務用戶端時，您可以設定預設引擎類型。SDK 會管理產生的 HTTP 用戶端引擎，並在不再需要時自動將其關閉。

下列範例顯示在 DynamoDB 用戶端初始化期間 HTTP 用戶端的組態。

匯入

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

Code

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

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

進階組態

預設 HTTP 組態適用於大多數使用案例。對於某些進階使用案例，例如高輸送量環境，下列進階組態選項提供額外的功能：

指定 HTTP 引擎類型

若要指定非預設 HTTP 引擎類型，或自訂特定 HTTP 引擎類型的組態，您可以將其他參數傳遞至 `httpClient` 以指定引擎類型。

下列範例指定[OkHttpEngine]您可以用來設定 [maxConcurrencyPerHost] 屬性的 。

匯入

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

Code

```
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.
}
```

引擎類型的可能值為 `OkHttpEngine`、 [OkHttp4Engine]和 [CrtHttpEngine]。

若要使用 HTTP 引擎特定的組態參數，您必須新增引擎做為編譯時間相依性。對於 `OkHttpEngine`，您可以使用 Gradle 新增下列相依性。

（您可以導覽至 *X.Y.Z* 連結，以查看可用的最新版本。）

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

```
implementation("aws.smithy.kotlin:http-client-engine-okhttp")
```

對於 CrtHttpEngine，新增下列相依性。

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

使用 OkHttp4Engine

OkHttp4Engine 如果您無法使用預設 ，請使用 OkHttpEngine。smithy-kotlin GitHub 儲存庫包含有關如何設定和使用 的資訊OkHttp4Engine。

使用明確的 HTTP 用戶端

當您使用明確的 HTTP 用戶端時，您必須對其生命週期負責，包括在不再需要它時將其關閉。只要使用任何服務用戶端，HTTP 用戶端必須至少存活。

下列程式碼範例顯示程式碼，可讓 HTTP 用戶端在DynamoDbClient作用中時保持運作狀態。use 函數可確保 HTTP 用戶端正確關閉。

匯入

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

Code

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

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

```
}
```

閒置連線監控

> **Important**
>
> OkHttp 引擎的連線閒置輪詢功能 (connectionIdlePollingInterval) 已由自動連線失敗重試 () 取代retryOnConnectionFailure。連線閒置輪詢將在 SDK 1.7 版中棄用，並在 SDK 1.8 版中移除。如需詳細資訊，請參閱相關的 GitHub 開發文章。

OkHttpEngine 提供用於監控閒置連線以進行遠端關閉的connectionIdlePollingInterval組態選項。此功能會偵測服務何時具有仍在連線集區中的關閉連線，以防止後續請求發生錯誤。

當 connectionIdlePollingInterval 設定為非空值時，引擎會輪詢釋放回連線集區的連線。輪詢程序會執行封鎖讀取，並將通訊端逾時設定為指定的間隔。當引擎從集區取得連線，或當連線被移出並關閉時，輪詢會自動取消。

當此值為 null（預設值）時，輪詢會停用。集區中遠端關閉的閒置連線可能會在後續呼叫時遇到錯誤。

> **Note**
>
> 由於輪詢迴圈使用封鎖讀取，引擎呼叫來取得或關閉連線可能會延遲與connectionIdlePollingInterval間隔一樣多。選擇間隔的低值表示 SDK 可以更快地取得連線，同時犧牲較高的閒置資源用量。

匯入

```
import aws.sdk.kotlin.services.s3.S3Client
import aws.smithy.kotlin.runtime.http.engine.okhttp.OkHttpEngine
import kotlin.time.Duration.Companion.milliseconds
```

Code

```
S3Client.fromEnvironment {
    httpEngine(OkHttpEngine) {
        connectionIdlePollingInterval = 50.milliseconds
    }
```

```
}.use { s3 ->
  // Use the Amazon S3 client
}
```

使用 HTTP 代理

若要使用AWS透過代理伺服器存取適用於 Kotlin 的 AWS SDK，您可以設定 JVM 系統屬性或環境變數。如果兩者都提供，則 JVM 系統屬性優先。

使用 JVM 系統屬性

SDK 會尋找 JVM 系統屬性 `https.proxyHost`、 `https.proxyPort`和 `http.nonProxyHosts`。如需這些常見 JVM 系統屬性的詳細資訊，請參閱 Java 文件中的聯網和代理。

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

使用環境變數

SDK 會尋找 `https_proxy`、 `http_proxy`和 `no_proxy`環境變數（以及每個變數的大寫版本）。

```
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
```

在 EC2 執行個體上使用代理

如果您在使用連接的 IAM 角色啟動的 EC2 執行個體上設定代理，請務必排除用於存取執行個體中繼資料的地址。若要這樣做，請將 `http.nonProxyHosts` JVM 系統屬性或`no_proxy`環境變數設定為執行個體中繼資料服務的 IP 地址，也就是 `169.254.169.254`。此地址不會改變。

```
export no_proxy=169.254.169.254
```

HTTP 攔截器

您可以使用攔截器來勾結 API 請求和回應的執行。攔截器是開放式機制，其中 SDK 會呼叫您編寫的程式碼，將行為注入請求/回應生命週期。如此一來，您就可以修改處理中的請求、除錯請求處理、檢視例外狀況等等。

下列範例顯示簡單的攔截器，在輸入重試迴圈之前，會將額外的標頭新增至所有傳出請求。

```
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()
    }
}
```

如需詳細資訊和可用的攔截掛鉤，請參閱攔截器界面。

攔截器註冊

您可以在建構服務用戶端或覆寫特定操作集的組態時註冊攔截器。

所有服務用戶端操作的攔截器

下列程式碼會將AddHeader執行個體新增至建置器的攔截器屬性。此新增會在輸入重試迴圈之前，將x-foo-version標頭新增至所有操作。

```
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 { ... }
```

僅特定操作的攔截器

透過使用 withConfig擴充功能，您可以覆寫任何服務用戶端的一或多個操作的服務用戶端組態。使用此功能，您可以為操作子集註冊其他攔截器。

下列範例會覆寫 use 延伸模組內操作的s3執行個體組態。在 上呼叫的操作同時s3Scoped包含 x-foo-version和 x-bar-version標頭。

```
// '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.
}
```

強制執行最低 TLS 版本

使用適用於 Kotlin 的 AWS SDK，您可以在連線至服務端點時設定最低 TLS 版本。SDK 提供不同的組態選項。依最高到最低優先順序排序，選項包括：

- 明確設定 HTTP 引擎
- 設定 `sdk.minTls` JVM 系統屬性
- 設定`SDK_MIN_TLS`環境變數

設定 HTTP 引擎

當您為服務用戶端指定非預設 HTTP 引擎時，您可以設定 `tlsContext.minVersion` 欄位。

下列範例會設定 HTTP 引擎和任何使用它來至少使用 TLS v1.2 的服務用戶端。

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

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

設定 **`sdk.minTls`** JVM 系統屬性

您可以設定 `sdk.minTls` JVM 系統屬性。當您使用系統屬性集啟動應用程式時，由 建構的所有 HTTP 適用於 Kotlin 的 AWS SDK引擎預設會使用指定的最低 TLS 版本。不過，您可以在 HTTP 引擎組態中明確覆寫此項目。允許的值為：

- TLS_1_0

- TLS_1_1
- TLS_1_2
- TLS_1_3

設定**SDK_MIN_TLS**環境變數

您可以設定SDK_MIN_TLS環境變數。當您使用環境變數集啟動應用程式時，除非被另一個選項覆寫，否則由 建構的所有 HTTP 引擎都會適用於 Kotlin 的 AWS SDK使用指定的最低 TLS 版本。

允許的值為：

- TLS_1_0
- TLS_1_1
- TLS_1_2
- TLS_1_3

中的重試次數適用於 Kotlin 的 AWS SDK

呼叫AWS 服務偶爾會傳回未預期的例外狀況。如果重試呼叫，某些類型的錯誤可能會成功，例如限流或暫時性錯誤。

此頁面說明如何自動適用於 Kotlin 的 AWS SDK處理重試，以及如何為您的應用程式自訂重試行為。

了解重試行為

下列各節說明 SDK 如何決定何時重試請求，以及哪些例外狀況被視為可重試。

預設重試組態

根據預設，每個服務用戶端都會自動設定標準重試策略。預設組態會嘗試最多失敗三次的呼叫（初始嘗試加上兩次重試）。每個呼叫之間的介入延遲會設定為指數退避和隨機抖動，以避免重試風暴。此組態適用於大多數使用案例，但在某些情況下可能不適合，例如高輸送量系統。

開發套件只會在可重試的錯誤上重試。可重試錯誤的範例包括通訊端逾時、服務端調節、並行或樂觀鎖定失敗，以及暫時性服務錯誤。遺失或無效的參數、身分驗證/安全錯誤，以及錯誤組態例外狀況都視為無法重試。

您可以透過設定最大嘗試次數、延遲和退避，以及字符儲存貯體組態來自訂標準重試策略。

哪些例外狀況可以重試？

適用於 Kotlin 的 AWS SDK使用預先設定的重試政策，決定哪些例外狀況可以重試。服務用戶端組態具有retryPolicy屬性，可指定套用至重試的政策。如果未指定自訂值，則預設值為 AwsRetryPolicy。

下列例外狀況由 判斷為可重試AwsRetryPolicy：

依錯誤碼重試

任何ServiceException具有 sdkErrorMetadata.errorCode的 ：

- BandwidthLimitExceeded
- EC2ThrottledException
- IDPCommunicationError
- LimitExceededException
- PriorRequestNotComplete
- ProvisionedThroughputExceededException
- RequestLimitExceeded
- RequestThrottled
- RequestThrottledException
- RequestTimeout
- RequestTimeoutException
- SlowDown
- ThrottledException
- Throttling
- ThrottlingException
- TooManyRequestsException
- TransactionInProgressException

可透過 HTTP 狀態碼重試

任何ServiceException具有 sdkErrorMetadata.statusCode的 ：

- 500 （內部服務錯誤）
- 502 （無效的閘道）

- 503 （服務無法使用）
- 504 （閘道逾時）

依錯誤類型重試

任何ServiceException具有 sdkErrorMetadata.errorType的 ：

- ErrorType.Server （例如內部服務錯誤）
- ErrorType.Client （例如無效的請求、找不到資源、存取遭拒等）

可由 SDK 中繼資料重試

下列任何SdkBaseException位置：

- sdkErrorMetadata.isRetryable 是 true（例如用戶端逾時、聯網/通訊端錯誤等）
- sdkErrorMetadata.isThrottling 是 true（例如在短時間內提出太多請求）

如需每個服務用戶端可能擲回之例外狀況的完整清單，請參閱服務特定的 API 參考文件。

檢查例外狀況是否可重試

若要判斷 SDK 是否將例外狀況視為可重試，請檢查抓取例外狀況的 isRetryable 屬性：

```
try {
    dynamoDbClient.putItem {
        tableName = "MyTable"
        item = mapOf("id" to AttributeValue.S("123"))
    }
} catch (e: SdkBaseException) {
    println("Exception occurred: ${e.message}")

    if (e.sdkErrorMetadata.isRetryable) {
        println("This exception is retryable - SDK will automatically retry")
        println("If you're seeing this, retries may have been exhausted")
    } else {
        println("This exception is not retryable - fix the underlying issue")

        // Common non-retryable scenarios.
        when {
            e.message?.contains("ValidationException") == true ->
```

```
                println("Check your request parameters")
            e.message?.contains("AccessDenied") == true ->
                println("Check your IAM permissions")
            e.message?.contains("ResourceNotFound") == true ->
                println("Verify the resource exists")
        }
    }
}
```

當重試失敗時，哪些例外狀況會到達您的程式碼

當開發套件的重試機制無法解決問題時，您的應用程式程式碼會擲回例外狀況。了解這些例外狀況類型可協助您實作適當的錯誤處理。這些不是觸發重試的例外狀況，這些都是由 SDK 在內部處理。

當重試用盡或停用時，您的程式碼會擷取以下類型的例外狀況：

重試耗盡後的服務例外狀況

當所有重試嘗試都失敗時，您的程式碼會擷取導致最後一次重試嘗試失敗的最終服務例外狀況 (的子類別`AwsServiceException`)。這可能是調節錯誤、伺服器錯誤，或 SDK 無法透過重試解決的其他服務特定例外狀況。

重試耗盡後的網路例外狀況

當網路問題在所有重試嘗試中持續存在時，您的程式碼會擷取`ClientException`執行個體是否有連線逾時、DNS 解析失敗，以及開發套件無法解決的其他連線問題。

使用下列模式來處理應用程式中的這些例外狀況：

```
try {
    s3Client.getObject {
        bucket = "amzn-s3-demo-bucket"
        key = "my-key"
    }
} catch (e: AwsServiceException) {
    // Service-side errors that persisted through all retries.
    println("Service error after retries: ${e.errorDetails?.errorCode} - ${e.message}")

    // Handle specific service errors that couldn't be resolved.
    if (e.errorDetails?.errorCode == "ServiceQuotaExceededException" ||
        e.errorDetails?.errorCode == "ThrottlingException") {
        println("Rate limiting persisted - consider longer delays or quota increase")
    }
```

```
} catch (e: ClientException) {
    // Client-side errors (persistent network issues, DNS resolution failures, etc.)
    println("Client error after retries: ${e.message}")
}
```

自訂重試行為

下列各節說明如何針對特定使用案例自訂 SDK 的重試行為。

設定最大嘗試次數

您可以在用戶端建構期間自訂 retryStrategy DSL 區塊中的預設最大嘗試次數 (3)。

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

使用上一個程式碼片段中顯示的 DynamoDB 服務用戶端，軟體開發套件會嘗試失敗最多五次的 API 呼叫（初次嘗試加上四次重試）。

您可以完全停用自動重試，方法是將最大嘗試次數設定為一個，如下列程式碼片段所示。

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

設定延遲和退避

如果需要重試，預設重試策略會在進行後續嘗試之前等待。第一次重試的延遲很小，但之後重試會呈指數增長。延遲的最大數量會受到限制，使其不會太大。

最後，隨機抖動會套用至所有嘗試之間的延遲。抖動有助於減輕可能導致重試風暴的大型機群的影響。（如需指數退避和抖動的深入討論，請參閱此AWS架構部落格文章。）

延遲參數可在 delayProvider DSL 區塊中設定。

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

```
        delayProvider {
            initialDelay = 100.milliseconds
            maxBackoff = 5.seconds
        }
    }
}
```

使用上一個程式碼片段中顯示的組態，用戶端會將第一次重試嘗試延遲最多 100 毫秒。任何重試嘗試之間的時間上限為 5 秒。

下列參數可用於調校延遲和退避。

參數	預設值	說明
`initialDelay`	10 毫秒	第一次重試的延遲量上限。套用抖動時，實際延遲量可能較少。
`jitter`	1.0 （全抖動）	隨機減少計算延遲的最大振幅。預設值 1.0 表示計算的延遲可以減少到 100% 的任何數量 （例如，低至 0)。值為 0.5 表示計算的延遲最多可減少一半。因此，最大延遲 10 毫秒可以減少到 5 毫秒到 10 毫秒之間的任何位置。值 0.0 表示不會套用抖動。 **Important** #抖動組態是一項進階功能。通常不建議自訂此行為。
`maxBackoff`	20 秒	套用至任何嘗試的最大延遲量。設定此值會限制後續嘗試之間發生的指數成長，並防止計算出的最大值過大。此參數

參數	預設值	說明
		會限制套用抖動之前的計算延遲。如果套用，抖動可能會進一步減少延遲。
`scaleFactor`	1.5	將增加後續最大延遲的指數基礎。例如，假設 `initialDelay` 為 10 毫秒，而 `scaleFactor` 為 1.5，則會計算下列最大延遲： • 重試 1：10ms × 1.50 = 10ms • 重試 2：10ms × 1.51 = 15ms • 重試 3：10ms × 1.52 = 22.5ms • 重試 4：10ms × 1.53 = 33.75ms 套用抖動時，每個延遲的實際數量可能較少。

設定重試權杖儲存貯體

您可以調整預設字符儲存貯體組態，進一步修改標準重試策略的行為。重試字符儲存貯體有助於減少不太可能成功或可能需要更多時間才能解決的重試，例如逾時和限流失敗。

> **Important**
>
> 字符儲存貯體組態是一項進階功能。通常不建議自訂此行為。

每次重試嘗試（選擇性地包含初次嘗試）都會從字符儲存貯體減少一些容量。減少的數量取決於嘗試的類型。例如，重試暫時性錯誤可能很便宜，但重試逾時或限流錯誤可能更昂貴。

成功嘗試會將容量傳回至儲存貯體。儲存貯體可能不會增加到超過容量上限，也不會減少到零以下。

根據useCircuitBreakerMode設定的值， 嘗試將容量減少到零以下會導致下列其中一個結果：

- 如果設定為 TRUE，則會擲回例外狀況 – 例如，如果發生太多重試次數，而且不太可能成功執行更多重試次數。
- 如果設定為 FALSE，則會有延遲 – 例如，延遲直到儲存貯體再次有足夠的容量。

> **Note**
>
> 當斷路器啟用 （權杖儲存貯體達到零容量） 時，軟體開發套件會擲回ClientException具有「超過重試容量」訊息的 。這是用戶端例外狀況，而不是 AwsServiceException，因為它源自開發套件的重試邏輯，而不是AWS服務。例外狀況會立即擲回，而不會嘗試操作，有助於防止在服務中斷期間重試風暴。

權杖儲存貯體參數可在 tokenBucket DSL 區塊中設定：

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

下列參數可用於調校重試字符儲存貯體：

參數	預設值	說明
initialTryCost	0	初始嘗試從儲存貯體減少的數量。預設值 0 表示不會減少任何容量，因此初始嘗試不會停止或延遲。
initialTrySuccessIncrement	1	初始嘗試成功時增加容量的數量。

參數	預設值	說明
maxCapacity	500	字符儲存貯體的最大容量。可用的字符數量不能超過此數量。
refillUnitsPerSecond	0	每秒重新新增至儲存貯體的容量。值為 0 表示不會自動新增任何容量。（例如，只有成功嘗試會導致容量增加）。值 0 useCircuitBreakerMode 必須為 TRUE。
retryCost	5	暫時性失敗後嘗試從儲存貯體減少的數量。如果嘗試成功，相同的數量會重新遞增回儲存貯體。
timeoutRetryCost	10	逾時或限流失敗後嘗試從儲存貯體減少的數量。如果嘗試成功，相同的數量會重新遞增回儲存貯體。
useCircuitBreakerMode	TRUE	決定當嘗試減少容量會導致儲存貯體的容量低於零時的行為。當 TRUE 時，字符儲存貯體會擲回例外狀況，表示不再存在重試容量。FALSE 時，字符儲存貯體會延遲嘗試，直到重新填充足夠的容量為止。

如需重試案例期間擲回的例外狀況類型詳細資訊，包括斷路器例外狀況，請參閱 the section called “當重試失敗時，哪些例外狀況會到達您的程式碼”。

設定自適應重試

作為標準重試策略的替代方案，自適應重試策略是一種進階方法，可尋求理想的請求率，以將限流錯誤降至最低。

> **Important**
>
> 自適應重試是一種進階重試模式。通常不建議使用此重試策略。

自適應重試包含標準重試的所有功能。它會新增用戶端速率限制器，測量節流請求相較於非節流請求的速率。它也會限制流量以嘗試保持在安全頻寬內，理想情況下會導致零限流錯誤。

速率會即時適應不斷變化的服務條件和流量模式，並可能相應地增加或減少流量速率。最重要的是，速率限制器可能會延遲高流量案例中的初始嘗試。

您可以為 `retryStrategy`方法提供額外的參數，以選取自適應重試策略。速率限制器參數可在 `rateLimiter DSL 區塊`中設定。

```
val dynamoDb = DynamoDbClient.fromEnvironment {
    retryStrategy(AdaptiveRetryStrategy) {
        maxAttempts = 10
        rateLimiter {
            minFillRate = 1.0
            smoothing = 0.75
        }
    }
}
```

> **Note**
>
> 自適應重試策略假設用戶端針對單一資源（例如，一個 DynamoDB 資料表或一個 Amazon S3 儲存貯體）運作。
> 如果您將單一用戶端用於多個資源，當用戶端存取所有其他資源時，與一個資源相關聯的限流或中斷會導致延遲增加和失敗。當您使用自適應重試策略時，我們建議您為每個資源使用單一用戶端。

可觀測性

可觀測性是從系統發出的資料推斷系統目前狀態的程度。發出的資料通常稱為遙測。

適用於 Kotlin 的 AWS SDK可以提供所有三種常見的遙測訊號：指標、追蹤和日誌。您可以連接 `TelemetryProvider`，將遙測資料傳送至可觀測性後端（例如 AWS X-Ray或 Amazon CloudWatch），然後對其採取行動。

根據預設，在 SDK 中只會啟用記錄，並停用其他遙測訊號。本主題說明如何啟用和設定遙測輸出。

> **Important**
>
> TelemetryProvider 目前是實驗性 API，必須選擇加入才能使用。

設定 TelemetryProvider

您可以針對所有服務用戶端或個別用戶端TelemetryProvider，在應用程式中全域設定 。下列範例使用假設getConfiguredProvider()函數來示範 TelemetryProvider API 操作。the section called “遙測供應商” 本節說明 SDK 提供的實作資訊。如果不支援提供者，您可以在 GitHub 上實作自己的支援或開啟功能請求。

設定預設全域遙測供應商

根據預設，每個服務用戶端都會嘗試使用全球可用的遙測提供者。如此一來，您可以設定一次提供者，所有用戶端都會使用它。在您執行個體化任何服務用戶端之前，這應該只完成一次。

若要使用全域遙測供應商，請先更新您的專案相依性，以新增遙測預設值模組，如下列 Gradle 程式碼片段所示。

（您可以導覽至 *X.Y.Z* 連結，以查看可用的最新版本。）

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

然後在建立服務用戶端之前設定全域遙測提供者，如下列程式碼所示。

```
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")
}
```

為特定服務用戶端設定遙測供應商

您可以使用特定的遙測提供者（全域提供者除外）來設定個別服務用戶端。如以下範例所示。

```
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")
}
```

指標

下表列出 SDK 發出的遙測指標。[設定遙測提供者](#)，讓指標可供觀察。

發出哪些指標？

指標名稱	單位	Type	屬性	說明
smithy.client.call.duration	s	直方圖	rpc.service、rpc.method	整體通話持續時間（包括重試）
smithy.client.call.attempts	{attempt}	MonotonCounter	rpc.service、rpc.method	個別操作的嘗試次數
smithy.client.call.errors	{error}	MonotonCounter	rpc.service、rpc.me	操作的錯誤數目

指標名稱	單位	Type	屬性	說明
			thod、 exception .type	
smithy.client.call .attempt_ duration	s	直方圖	rpc.servi ce、rpc.method	連線到服務、傳送請求，以及取回 HTTP 狀態碼和標頭所需的時間 （包括排入佇列等待傳送的時間）
smithy.client.call .resolve_ endpoint_ duration	s	直方圖	rpc.servi ce、rpc.method	解析請求的端點 （端點解析程式，而非 DNS) 所需的時間
smithy.client.call .serialization_dur ation	s	直方圖	rpc.servi ce、rpc.method	序列化訊息內文所需的時間
smithy.cl ient.call.deserial ization_duration	s	直方圖	rpc.servi ce、rpc.method	還原序列化訊息內文所需的時間
smithy.cl ient.call.auth.sig ning_duration	s	直方圖	rpc.servi ce、rpc.me thod、auth .scheme_id	簽署請求所需的時間
smithy.client.call .auth.res olve_iden tity_duration	s	直方圖	rpc.servi ce、rpc.me thod、auth .scheme_id	從身分提供者取得身分 （例如AWS憑證或承載字符） 所需的時間
smithy.client.http .connecti ons.acqui re_duration	s	直方圖		請求取得連線所需的時間

指標名稱	單位	Type	屬性	說明
smithy.client.http.connections.limit	{connection}	【非同步】UpDownCount		針對 HTTP 用戶端允許/設定的開啟連線上限
smithy.client.http.connections.usage	{connection}	【非同步】UpDownCount	狀態：閒置 | 已取得	連線集區的目前狀態
smithy.client.http.connections.uptime	s	直方圖		連線已開啟的時間量
smithy.client.http.requests.usage	{request}	【非同步】UpDownCount	狀態：已排入佇列 | 進行中	HTTP 用戶端請求並行的目前狀態
smithy.client.http.requests.queued_duration	s	直方圖		請求排入佇列並等待 HTTP 用戶端執行的時間
smithy.client.http.bytes_sent	根據	MonotonCounter	server.address	HTTP 用戶端傳送的位元組總數
smithy.client.http.bytes_received	根據	MonotonCounter	server.address	HTTP 用戶端收到的位元組總數

以下是欄描述：

- 指標名稱 – 發出的指標名稱。
- 單位 – 指標的度量單位。單位是以區分大小寫的 UCUM ("c/s") 表示法提供。
- 類型 – 用來擷取指標的檢測類型。
- 描述 – 指標測量內容的描述。
- 屬性 – 以 指標發出的一組屬性 （ 維度 ）。

記錄

會將 SLF4J 相容記錄器適用於 Kotlin 的 AWS SDK設定為遙測提供者LoggerProvider的預設值。使用 SLF4J 是一種抽象層，您可以在執行時間使用多個記錄系統中的任何一個。支援的記錄系統包括 Java Logging APIs、Log4j 2 和 Logback。

> **Warning**
>
> 我們建議您僅使用線路記錄進行除錯。（以下將討論線路記錄。）在生產環境中將其關閉，因為它可以記錄敏感資料，例如電子郵件地址、安全字符、API 金鑰、密碼和AWS Secrets Manager秘密。線路記錄會記錄完整的請求或回應，無需加密，即使是 HTTPS 呼叫也一樣。對於大型請求（例如將檔案上傳至 Amazon S3) 或回應，詳細的線路記錄也會大幅影響應用程式的效能。

Log4j 2 記錄組態範例

雖然可以使用任何 SLF4J相容的日誌程式庫，但此範例會使用 Log4j 2 從 JVM 程式中的 SDK 啟用日誌輸出：

梯度相依性

（您可以導覽至 *X.Y.Z* 連結，以查看可用的最新版本。）

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

Log4j 2 組態檔案

在log4j2.xmlresources目錄中建立名為 的檔案（例如，<project-dir>/src/main/resources)。將下列 XML 組態新增至 檔案：

```
<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>
```

此組態包含 pattern 屬性中的%X指標，可啟用 MDC （映射診斷內容） 記錄。

軟體開發套件會為每個操作新增下列 MDC 元素。

rpc

調用 RPC 的名稱，例如 S3.GetObject。

sdkInvocationId

由服務用戶端為 操作指派的唯一 ID。ID 會關聯與呼叫單一 操作相關的所有記錄事件。

指定線路層級訊息的日誌模式

根據預設，適用於 Kotlin 的 AWS SDK不會記錄線路層級訊息，因為它們可能包含來自 API 請求和回應的敏感資料。不過，有時您需要此層級的詳細資訊以進行偵錯。

使用 Kotlin 開發套件，您可以在程式碼中或使用環境設定來啟用下列項目的偵錯訊息：

- HTTP 請求
- HTTP 回應

日誌模式的後端是位元欄位，其中每個位元都是旗標 （模式），而值是累加的。您可以結合一個請求模式和一個回應模式。

在程式碼中設定日誌模式

若要選擇加入其他記錄，請在建構服務用戶端時設定 logMode 屬性。

下列範例示範如何啟用 請求記錄 （含內文） 和 回應 （不含內文）。

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

// ...

val client = DynamoDbClient {
    // ...
```

```
    logMode = LogMode.LogRequestWithBody + LogMode.LogResponse
}
```

在服務用戶端建構期間設定的日誌模式值， 會覆寫從環境設定的任何日誌模式值。

從環境設定日誌模式

若要針對程式碼中未明確設定的所有服務用戶端全域設定日誌模式，請使用下列其中一項：

- JVM 系統屬性： `sdk.logMode`
- 環境變數： `SDK_LOG_MODE`

可使用以下不區分大小寫的值：

- `LogRequest`
- `LogRequestWithBody`
- `LogResponse`
- `LogResponseWithBody`

若要使用 環境的設定建立合併日誌模式，您可以使用管道 (|) 符號分隔值。

例如，下列範例會設定與先前範例相同的日誌模式。

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

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

> **Note**
>
> 您還必須設定相容的 SLF4J 記錄器，並將記錄層級設定為 DEBUG，以啟用線路層級記錄。

遙測供應商

開發套件目前支援 OpenTelemetry (OTel) 做為供應商。軟體開發套件未來可能會提供額外的遙測供應商。

主題

- 設定 OpenTelemetry 型遙測供應商

設定 OpenTelemetry 型遙測供應商

適用於 Kotlin 的 SDK 提供 OpenTelemetry 支援的`TelemetryProvider`介面實作。

先決條件

更新您的專案相依性以新增 OpenTelemetry 供應商，如下列 Gradle 程式碼片段所示。您可以導覽至 *X.Y.Z* 連結，以查看可用的最新版本。

```
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")
}
```

設定軟體開發套件

下列程式碼會使用 OpenTelemetry 遙測提供者來設定服務用戶端。

```
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**
>
> 有關如何設定 OpenTelemetry SDK 的討論不在本指南的範圍內。OpenTelemetry Java 文件包含各種方法的組態資訊：手動、透過 Java 代理程式自動或（選用）收集器。

Resources

下列資源可協助您開始使用 OpenTelemetry。

- AWS Distro for OpenTelemetry -AWS OTeL Distro 首頁
- aws-otel-java-instrumentation -AWS Distro for OpenTelemetry Java 檢測程式庫
- aws-otel-lambda 受AWS管 OpenTelemetry Lambda 層
- aws-otel-collector -AWS Distro for OpenTelemetry Collector
- AWS可觀測性最佳實務 - 特定於 的可觀測性一般最佳實務AWS

覆寫服務用戶端組態

建立服務用戶端之後，服務用戶端會針對所有操作使用固定組態。不過，有時您可能需要覆寫一或多個特定操作的組態。

每個服務用戶端都有`withConfig`擴充功能，讓您可以修改現有組態的副本。`withConfig` 延伸項目會傳回具有修改後組態的新服務用戶端。原始用戶端獨立存在，並使用其原始組態。

下列範例顯示建立呼叫兩個操作的`S3Client`執行個體。

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

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

下列程式碼片段顯示如何覆寫單一`listObjectV2`操作的組態。

```
s3.withConfig {
```

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

s3 用戶端上的操作呼叫會使用建立用戶端時指定的原始組態。其組態包含 us-west-2 region 區域的請求記錄和 。

overriddenS3 用戶端上的listObjectsV2調用使用與原始s3用戶端相同的設定，但 區域除外，現在為 eu-central-1。

覆寫用戶端的生命週期

在先前的範例中，s3用戶端和overriddenS3用戶端彼此獨立。只要操作保持開啟狀態，就可以在任一用戶端上叫用操作。每個 使用單獨的組態，但除非這些組態也被覆寫，否則它們可以共用基礎資源（例如 HTTP 引擎）。

您可以分別關閉具有覆寫組態的用戶端和原始用戶端。您可以在關閉原始用戶端之前或之後，關閉具有覆寫組態的用戶端。除非您需要長時間使用具有覆寫組態的用戶端，否則建議您使用 use方法包裝其生命週期。use 方法可確保在發生例外狀況時關閉用戶端。

用戶端之間共用的資源

當您使用 建立服務用戶端時withConfig，它可能會與原始用戶端共用資源。相反地，當您使用 fromEnvironment 建立用戶端或明確設定用戶端時，用戶端會使用獨立資源。除非在 withConfig區塊中覆寫，否則會共用 HTTP 引擎和登入資料提供者等資源。

由於每個用戶端的生命週期都是獨立的，共用資源會保持開啟和可用，直到最後一個用戶端關閉為止。因此，當您不再需要覆寫的服務用戶端時，請務必將其關閉。這可防止共用資源保持開啟並耗用系統資源，例如記憶體、連線和 CPU 週期。

下列範例顯示共用和獨立資源。

s3 和 overriddenS3用戶端共用相同的登入資料提供者執行個體，包括其快取組態。如果快取值仍是s3用戶端所進行呼叫的目前值，則透過overriddenS3重複使用登入資料進行的呼叫。

HTTP 引擎不會在兩個用戶端之間共用。每個用戶端都有獨立的 HTTP 引擎，因為它在withConfig呼叫中遭到覆寫。

```
val s3 = S3Client.fromEnvironment {
```

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

s3.listBuckets { ... }

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

使用 SDK

本節提供使用 所需的基本資訊 適用於 Kotlin 的 AWS SDK。

主題

- 提出請求
- Coroutines
- 串流操作
- 分頁
- 等待程式
- 錯誤處理
- 預先簽章請求
- 疑難排解常見問答集
- 中的模擬 適用於 Kotlin 的 AWS SDK

提出請求

使用 服務用戶端向 提出請求 AWS 服務。 適用於 Kotlin 的 AWS SDK 提供符合類型安全建置器模式的網域特定語言 (DSLs) 來建立請求。請求的巢狀結構也可以透過其 DSLs存取。

下列範例示範如何建立 Amazon DynamoDB createTable 操作輸入：

```
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)
```

服務介面 DSL 過載

服務用戶端界面上的每個非串流操作都有 DSL 超載，因此您不需要建立個別的請求。

使用過載函數建立 Amazon Simple Storage Service (Amazon S3) 儲存貯體的範例：

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

這相當於：

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

s3client.createBucket(request)
```

沒有必要輸入的請求

不需要輸入的操作可以呼叫，而無需傳遞請求物件。這通常適用於清單類型操作，例如 Amazon S3 `listBuckets` API 操作。

例如，以下三個陳述式是相等的：

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

Coroutines

根據預設， 適用於 Kotlin 的 AWS SDK 是非同步的。適用於 Kotlin 的 SDK 對所有操作使用 `suspend`函數，這些函數旨在從 coroutine 呼叫。

如需 coroutine 的更深入指南，請參閱官方 Kotlin 文件。

提出並行請求

非同步 coroutine 建置器可用於啟動您關心結果的並行請求。 `async`會傳回延遲，代表輕量、非封鎖的未來，代表稍後提供結果的承諾。

如果您不在意結果（僅 操作已完成），您可以使用啟動 coroutine builder。 `launch` 概念上類似於 `async`。差別在於啟動會傳回任務，且不會包含任何產生的值，而 會`async`傳回 `Deferred`。

以下是使用 headObject 操作向 Amazon S3 提出並行請求的範例，以取得兩個金鑰的內容大小：

```
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")

}
```

提出封鎖請求

若要從不使用 coroutine 的現有程式碼進行服務呼叫，並實作不同的執行緒模型，您可以使用 runBlocking coroutine builder。不同執行緒模型的範例是使用 Java 的傳統執行器/未來方法。如果您要混合 Java 和 Kotlin 程式碼或程式庫，您可能需要使用此方法。

顧名思義，此`runBlocking`建置器會啟動新的 coroutine，並封鎖目前的執行緒，直到完成為止。

> **Warning**
>
> `runBlocking` 通常不應從 coroutine 中使用。它旨在將一般封鎖程式碼橋接到以暫停樣式寫入的程式庫（例如在主要函數和測試中）。

串流操作

在 中 適用於 Kotlin 的 AWS SDK，二進位資料 （ 串流 ） 表示為ByteStream類型，這是抽象的唯讀位元組串流。

串流回應

使用二進位串流 （ 例如 Amazon Simple Storage Service (Amazon S3) GetObject API 操作 ） 的回應處理方式與其他方法不同。這些方法採用處理回應的 lambda 函數，而不是直接傳回回應。這會限制對函數的回應範圍，並簡化呼叫者和 SDK 執行時間的生命週期管理。

在 lambda 函數傳回後，會釋出任何資源，例如基礎 HTTP 連線。(ByteStream不應在 Lambda 傳回後存取 ，也不應從關閉中移出。) 呼叫的結果是 lambda 傳回的任何內容。

下列程式碼範例顯示接收 lambda 參數的 getObject 函數，該參數會處理回應。

```
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")
```

對於常見的使用方式， ByteStream類型具有下列延伸：

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

所有這些都是在 aws.smithy.kotlin.runtime.content套件中定義。

串流請求

若要提供 ByteStream，也有數種便利方法，包括下列項目：

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

所有這些都是在 aws.smithy.kotlin.runtime.content套件中定義。

下列程式碼範例顯示使用ByteStream便利方法，在建立 PutObjectRequest 時提供內文屬性：

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

分頁

當承載太大而無法在單一回應中傳回時，許多 AWS 操作會傳回分頁結果。 適用於 Kotlin 的 AWS SDK 包含服務用戶端界面的延伸，可為您自動分頁結果。您只需要撰寫處理結果的程式碼。

分頁會公開為 Flow<T>，因此您可以利用 Kotlin 的非同步集合的慣用轉換（例如 map、 filter和 take)。例外情況是透明的，這使得錯誤處理感覺像是一般 API 呼叫，而取消遵守一般合作取消 coroutine。如需詳細資訊，請參閱官方指南中的流程和流程例外狀況。

> **Note**
>
> 下列範例使用 Amazon S3。不過，對於具有一或多個分頁 APIs 的任何服務，這些概念都相同。所有分頁副檔名都定義在aws.sdk.kotlin.services.<service>.paginators套件中（例如 aws.sdk.kotlin.services.dynamodb.paginators)。

下列程式碼範例示範如何從 listObjectsV2Paginated 函數呼叫處理分頁回應。

匯入

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

Code

```
val s3 = S3Client.fromEnvironment()
val req = ListObjectsV2Request {
    bucket = "amzn-s3-demo-bucket"
    maxKeys = 1
}

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

等待程式

等待程式是一種用戶端抽象，用於輪詢資源，直到達到所需的狀態，或直到確定資源不會進入所需的狀態為止。這是在使用最終一致的服務時常見的任務，例如 Amazon Simple Storage Service (Amazon S3)，或是非同步建立資源的服務，例如 Amazon EC2。

編寫邏輯以持續輪詢資源的狀態可能會很麻煩且容易出錯。等待程式的目標是將此責任移出客戶程式碼，並移至 適用於 Kotlin 的 AWS SDK，其對 AWS 操作的時間層面有深入的了解。

> **Note**
>
> 下列範例使用 Amazon S3。不過，對於 AWS 服務 任何已定義一或多個等待程式的 ，這些概念都相同。所有延伸項目都會在`aws.sdk.kotlin.services.<service>.waiters`套件中定義 （例如 `aws.sdk.kotlin.services.dynamodb.waiters`)。他們也遵循標準命名慣例 (`waitUntil<Condition>`)。

下列程式碼範例顯示使用等待程式函數，可讓您避免寫入輪詢邏輯。

匯入

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

Code

```
val s3 = S3Client.fromEnvironment()

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

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

// The bucket now exists.
```

> **Note**
>
> 每個等待方法都會傳回執行個體，該Outcome執行個體可用來取得對應至所需條件的最終回應。結果也包含其他詳細資訊，例如嘗試達到所需狀態的次數。

錯誤處理

了解 適用於 Kotlin 的 AWS SDK 拋出例外狀況的方式和時間，對於使用 SDK 建置高品質應用程式至關重要。以下章節說明開發套件會擲回的不同例外狀況案例，以及如何正確處理這些狀況。

服務例外狀況

最常見的例外狀況是 AwsServiceException，所有服務特定的例外狀況 （例如 S3Exception) 都會從中繼承。此例外狀況代表來自 的錯誤回應 AWS 服務。例如，如果您嘗試終止不存在的 Amazon EC2 執行個體，Amazon EC2 會傳回錯誤回應。錯誤回應詳細資訊會包含在擲出AwsServiceException的 中。

當您遇到 時AwsServiceException，這表示您的請求已成功傳送到 ， AWS 服務 但無法處理。這可能是因為請求參數中的錯誤，或因為服務端的問題。

用戶端例外狀況

ClientException 表示在嘗試傳送請求至 或嘗試剖析回應 AWS 時，用戶端程式碼內 適用於 Kotlin 的 AWS SDK 發生問題 AWS。ClientException 通常比 更嚴重，AwsServiceException表示主要問題是用戶端無法處理 的服務呼叫 AWS 服務。例如，ClientException如果 無法剖析來自服務的回應， 會 適用於 Kotlin 的 AWS SDK 擲回 。

錯誤中繼資料

每個服務例外狀況和用戶端例外狀況都有 sdkErrorMetadata 屬性。這是類型屬性包，可用於擷取有關錯誤的其他詳細資訊。

AwsErrorMetadata 類型直接有數個預先定義的擴充功能，包括但不限於下列項目：

- sdkErrorMetadata.requestId – 唯一的請求 ID
- sdkErrorMetadata.errorMessage – 人類可讀訊息 （ 通常與 相符Exception.message，但如果 服務的例外狀況不明，則可能包含更多資訊 ）
- sdkErrorMetadata.protocolResponse – 原始通訊協定回應

下列範例示範如何存取錯誤中繼資料。

```
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}")
}
```

預先簽章請求

您可以為某些 AWS API 操作預先簽署請求，以便其他發起人稍後可以使用該請求，而無需出示自己的登入資料。

例如，假設 Alice 可以存取 Amazon Simple Storage Service (Amazon S3) 物件，而且她想要暫時與 Bob 共用物件存取。Alice 可以產生預先簽章的GetObject請求來與 Bob 共用，以便他可以下載物件，而無需存取 Alice 的登入資料。

預先簽章基本概念

適用於 Kotlin 的 SDK 在服務用戶端上提供用於預先簽署請求的延伸方法。所有預先簽章的請求都需要代表已簽署請求有效時間長度的持續時間。持續時間結束後，預先簽章的請求會過期，並在執行時引發身分驗證錯誤。

下列程式碼顯示為 Amazon S3 建立預先簽章`GetObject`請求的範例。請求在建立後 24 小時內有效。

```
val s3 = S3Client.fromEnvironment()

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

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

`presignGetObject` 延伸項目方法會傳回 `HttpRequest` 物件。請求物件包含可叫用操作的預先簽章 URL。另一個呼叫者可以在不同的程式碼庫或程式設計語言環境中使用 URL （或整個請求）。

建立預先簽章的請求之後，請使用 HTTP 用戶端來叫用請求。叫用 HTTP GET 請求的 API 取決於 HTTP 用戶端。下列範例使用 Kotlin `URL.readText`方法。

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

進階預先簽章組態

在 SDK 中，每個可以預先簽署請求的方法都有過載，您可以用來提供進階組態選項，例如特定的簽署者實作或詳細的簽署參數。

下列範例顯示使用 CRT 簽署者變體並指定未來簽署日期的 Amazon S3 `GetObject`請求。

```
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
}
```

傳回的預先簽章請求 24 小時前為轉寄日期，在此之前無效。它在那之後 8 小時過期。

預先簽署 POST 和 PUT 請求

許多可預先簽章的操作只需要 URL，而且必須以 HTTP GET 請求執行。不過，某些操作會採用內文，在某些情況下，必須以 HTTP POST 或 HTTP PUT 請求以及標頭的形式執行。預先簽章這些請求與預先簽章 GET 請求相同，但叫用預先簽章請求更為複雜。

以下是預先簽署 S3 `PutObject`請求的範例：

```
val s3 = S3Client.fromEnvironment()

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

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

傳回的方法值`HttpRequest`為 ，`HttpMethod.PUT`並包含 URL 和標頭，該 URL 和標頭必須包含在 HTTP 請求的未來調用中。您可以將此請求傳遞給可在不同程式碼庫或程式設計語言環境中執行請求的發起人。

建立預先簽章的 POST 或 PUT 請求之後，請使用 HTTP 用戶端來叫用請求。叫用 POST 或 PUT 請求 URL 的 API 取決於使用的 HTTP 用戶端。下列範例使用 OkHttp HTTP 用戶端，並包含包含 的內文`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()
```

軟體開發套件可以預先簽署的操作

適用於 Kotlin 的 SDK 目前支援預先簽署以下需要使用相關聯 HTTP 方法呼叫的 API 操作。

AWS 服務	作業	SDK 擴充功能方法	使用 HTTP 方法
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

疑難排解常見問答集

當您 適用於 Kotlin 的 AWS SDK 在應用程式中使用 時，您可能會遇到本主題中列出的一些問題。使用下列建議來協助找出根本原因並解決錯誤。

如何修正「連線已關閉」問題？

您可能會遇到「連線關閉」問題，例如下列其中一種類型的例外狀況：

- `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)`

這些例外狀況表示從 SDK 到服務的 TCP 連線意外關閉或重設。您的主機、 AWS 服務或 NAT 閘道、代理或負載平衡器等中介方可能已關閉連線。

這些類型的例外狀況會自動重試，但仍然會出現在 SDK 日誌中，視您的記錄組態而定。如果將例外狀況擲入您的程式碼中，表示作用中的重試策略已耗盡其設定的限制，例如最大嘗試次數或重試權杖儲存貯體。如需重試策略的詳細資訊，請參閱本指南的 the section called “重試”一節。另請參閱the section called “為什麼在達到嘗試次數上限之前擲回例外狀況？”。

使用 OkHttpEngine 進行閒置連線監控

如果您使用 `OkHttpEngine`並經常遇到`IOException: unexpected end of stream on <URL>`例外狀況，請考慮啟用閒置連線監控。此功能會偵測遠端伺服器何時具有仍在連線集區中的封閉連線，這可以減少這些例外狀況的發生。

為什麼在達到嘗試次數上限之前擲回例外狀況？

有時您可能會看到預期會重試的例外狀況，但會改為擲回。在這些情況下，下列步驟可能有助於解決問題。

- 確認例外狀況可重試。有些例外狀況無法重試，例如表示服務請求格式不正確、缺少許可和不存在的資源等。開發套件不會自動重試這類例外狀況。如需如何檢查可重試例外狀況的資訊，請參閱 the section called “檢查例外狀況是否可重試”。
- 確認例外狀況已擲入您的程式碼中。有些例外狀況會在日誌訊息中顯示為資訊，但實際上不會擲入您的程式碼中。例如，由於 SDK 會自動執行多個backoff-and-retry週期，因此可能會記錄可重試的例外狀況，例如調節錯誤。只有在設定的重試設定未處理 SDK 操作時，才會擲回例外狀況。
- 驗證您設定的重試設定。如需重試策略和重試政策的詳細資訊，請參閱本指南的 the section called “重試”一節。確保您的程式碼使用您預期的設定或自動預設值。
- 請考慮調整您的重試設定。驗證先前項目但問題未解決後，您可以考慮調整重試設定。
  - 增加最大嘗試次數。根據預設， 操作的嘗試次數上限為 3 次。如果您發現這不夠，且預設設定仍然發生例外狀況，請考慮在用戶端組態中增加 `retryStrategy.maxAttempts` 屬性。如需詳細資訊，請參閱the section called “最大嘗試次數”。
  - 增加延遲設定。在基礎條件有機會解決之前，某些例外狀況可能會嘗試過快。如果您懷疑確實如此，請考慮增加用戶端組態中的 `retryStrategy.delayProvider.initialDelay`或 `retryStrategy.delayProvider.maxBackoff` 屬性。如需詳細資訊，請參閱the section called “延遲和退避”。
  - 停用斷路器模式。軟體開發套件預設會維護每個服務用戶端的字符儲存貯體。當 SDK 嘗試請求，但失敗並出現可重試的例外狀況時，字符計數會減少；當請求成功時，字符計數會遞增。

    根據預設，如果此字符儲存貯體達到剩餘 0 個字符，則電路會中斷。電路中斷後，開發套件會停用重試，且任何在第一次嘗試時失敗的目前和後續請求都會立即擲回例外狀況。開發套件會在初次

嘗試成功傳回足夠的容量給字符儲存貯體後重新啟用重試。此行為是刻意的，旨在防止在服務中斷和服務復原期間重試風暴。

如果您希望軟體開發套件繼續重試直到設定的嘗試次數上限，請考慮在用戶端組態中將 `retryStrategy.tokenBucket.useCircuitBreakerMode` 屬性設定為 false，以停用斷路器模式。將此屬性設定為 false 時，SDK 用戶端會等到字符儲存貯體達到足夠的容量，而不是放棄進一步的重試，這可能會在剩餘 0 個字符時導致例外狀況。

如何修正 **NoSuchMethodError**或 NoClassDefFoundError？

這些錯誤最常見的原因是相依性遺失或衝突。如需詳細資訊，請參閱the section called “如何解決相依性衝突？”。

我看到適用於 **NoClassDefFoundError**的 **okhttp3/coroutines/ExecuteAsyncKt**

這表示 OkHttp 的相依性問題。如需詳細資訊，請參閱the section called “解決應用程式中的 OkHttp 版本衝突”。

如何解決相依性衝突？

當您使用 時 適用於 Kotlin 的 AWS SDK，它需要特定 AWS 和第三方相依性才能正常運作。如果這些相依性在執行時間遺失或未預期的版本，您可能會看到 `NoSuchMethodError`或 等錯誤`NoClassDefFoundError`。這些相依性問題通常分為兩個群組：

- SDK/Smithy 相依性衝突
- 第三方相依性衝突

當您建置 Kotlin 應用程式時，可能會使用 Gradle 來管理相依性。在 SDK 服務用戶端上新增相依性到您的專案，會自動包含所有必要的相關相依性。不過，如果您的應用程式有其他相依性，它們可能會與 SDK 所需的相依性衝突。例如，開發套件依賴 OkHttp，這是您的應用程式可能也會使用的熱門 HTTP 用戶端。為了協助您發現這些衝突，Gradle 提供便利的任務，列出專案的相依性：

```
./gradlew dependencies
```

當您遇到相依性衝突時，您可能需要採取動作。您可以在本機命名空間中指定特定版本的相依性或影子相依性。Gradle 相依性解析是一個複雜的主題，在 Gradle 使用者手冊的下列章節中進行了討論：

- 了解相依性解析
- 相依性限制和衝突解決
- 對齊相依性版本

在專案中管理 SDK 和 Smithy 相依性

當您使用 SDK 時，請記住，其模組通常依賴其他版本編號相符的 SDK 模組。例如，`aws.sdk.kotlin:s3:1.2.3`取決於 `ws.sdk.kotlin:aws-http:1.2.3`，這取決於 `aws.sdk.kotlin:aws-core:1.2.3`，以此類推。

開發套件模組也會使用特定的 Smithy 模組版本。雖然 Smithy 模組版本不會與 SDK 版本編號同步，但它們必須符合 SDK 的預期版本。例如， `aws.sdk.kotlin:s3:1.2.3`可能取決於 `aws.smithy.kotlin:serde:1.1.1`，這取決於 `aws.smithy.kotlin:runtime-core:1.1.1`，以此類推。

若要避免相依性衝突，請將所有 SDK 相依性一起升級，並針對任何明確的 Smithy 相依性執行相同的動作。考慮使用我們的 Gradle 版本目錄來保持版本同步，並消除 SDK 和 Smithy 版本之間的映射猜測。

請記住，SDK/Smithy 模組中的次要版本更新可能包括重大變更，如我們的版本控制政策所述。在次要版本之間升級時，請仔細檢閱變更日誌並徹底測試執行時間行為。

解決應用程式中的 OkHttp 版本衝突

OkHttp 是開發套件預設在 JVM 上使用的熱門 HTTP 引擎。您的應用程式可能包含具有不同 OkHttp 版本的其他相依性或架構。這可能會導致`okhttp3`命名空間中`NoClassDefFoundError`類別的 ，例如 `okhttp/coroutines/ExecuteAsyncKt`或 `okhttp3/ConnectionListener`。發生這種情況時，您通常應該選擇較新的版本來解決衝突。為了協助您追蹤這些衝突的來源，Gradle 提供了有用的任務。您可以執行下列動作來列出所有相依性：

```
./gradlew dependencies
```

例如，如果 SDK 依賴 OkHttp，`5.0.0-alpha.14`而 Spring Boot 等其他相依性依賴 OkHttp`4.12.0`，則您應該使用 `5.0.0-alpha.14 version`。您可以使用 Gradle 中的`constraints`區塊來執行此操作：

```
dependencies {
    constraints {
```

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

或者，如果您必須使用 OkHttp 4.x，開發套件會提供 `OkHttp4Engine`。如需如何在程式碼`OkHttp4Engine`中設定 Gradle 和使用 的資訊，請參閱 文件。

中的模擬 適用於 Kotlin 的 AWS SDK

開發人員可以使用數個架構，透過 在測試中執行模擬 適用於 Kotlin 的 AWS SDK。本主題記錄一些架構所需的額外組態或特殊考量。

MockK

當您使用 MockK 模擬整個模組的延伸函數時，您需要執行額外的組態。在適用於 Kotlin 的 SDK 中，分頁程式、等待程式和預先簽章程式是延伸函數的範例，因此在模擬其行為時，您需要額外的組態。

您必須先呼叫 ，`mockkStatic("<MODULE_CLASS_NAME>")`才能設定您的模擬。一般而言，模組類別名稱為：

- 分頁程式： `aws.sdk.kotlin.services.<service>.paginators.PaginatorsKt`
- 等待程式： `aws.sdk.kotlin.services.<service>.waiters.WaitersKt`
- 預先簽章者： `aws.sdk.kotlin.services.<service>.presigners.PresignersKt`

例如，在以下包含模擬 `listBucketsPaginated`- 分頁器延伸函數 - 的測試中，我們新增 `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)
    }
```

如果沒有 mockkStatic，您會看到下列錯誤：

```
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)
```

如果前簽署者延伸函數沒有 mockkStatic，您可能會看到：

```
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)
```

所有範例成品

測試中的程式碼

```
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()
    }
}
```

測試類別

```
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 服務 使用 適用於 Kotlin 的 AWS SDK

本章包含如何使用 AWS 服務 適用於 Kotlin 的 SDK 來使用 的資訊。

內容

- 使用 使用 Amazon S3 適用於 Kotlin 的 AWS SDK
  - 使用檢查總和保護資料完整性
    - 上傳物件
      - 使用預先計算的檢查總和值
      - 分段上傳
    - 下載物件
      - 非同步驗證
  - 使用適用於 Kotlin 的 SDK 來使用 Amazon S3 多區域存取點
    - 使用多區域存取點
    - 使用物件和多區域存取點
- 使用 使用 DynamoDB 適用於 Kotlin 的 AWS SDK
  - 使用以 AWS 帳戶為基礎的端點
  - 使用 DynamoDB Mapper 將類別映射至 DynamoDB 項目 （開發人員預覽版）
    - DynamoDB Mapper 入門
      - 新增相依性
      - 建立和使用映射器
      - 使用類別註釋定義結構描述
      - 叫用 操作
        - 使用分頁回應
    - 設定 DynamoDB Mapper
      - 使用攔截器
        - 了解請求管道
        - 勾點
          - 唯讀勾點
          - 修改掛鉤
          - 執行順序

        - 範例組態
- 從註釋產生結構描述
    - 套用外掛程式
    - 設定 外掛程式
    - 註釋類別
        - 類別註釋
        - 屬性註釋
    - 定義自訂項目轉換器
- 手動定義結構描述
    - 在程式碼中定義結構描述
- 搭配 DynamoDB Mapper 使用次要索引
    - 定義次要索引的結構描述
    - 在 操作中使用次要索引
- 使用表達式
    - 在 操作中使用表達式
    - DSL 元件
        - 屬性
        - 等式和不等式
        - 範圍和集
        - 布林值邏輯
        - 函數和屬性
        - 排序金鑰篩選條件

使用 使用 Amazon S3 適用於 Kotlin 的 AWS SDK

Amazon Simple Storage Service for Kotlin SDK 的主要界面是 S3Client。使用 SDK 中的`S3Client`類似其他服務用戶端向 Amazon S3 提出請求。

協助您搭配 S3 使用 Kotlin 開發套件的資源包括：

- 適用於 S3 的 Kotlin SDK API 參考。
- S3 服務使用者指南和服務 API 參考。

下列主題提供使用 S3 之特定 Kotlin SDK APIs引導式程式碼範例。

主題

- 使用檢查總和保護資料完整性
- 使用適用於 Kotlin 的 SDK 來使用 Amazon S3 多區域存取點

使用檢查總和保護資料完整性

Amazon Simple Storage Service (Amazon S3) 可讓您在上傳物件時指定檢查總和。當您指定檢查總和時，它會與 物件一起存放，並且可以在下載物件時進行驗證。

當您傳輸檔案時，檢查總和可提供多一層的資料完整性。使用檢查總和，您可以透過確認收到的檔案符合原始檔案來驗證資料一致性。如需使用 Amazon S3 檢查總和的詳細資訊，請參閱 Amazon Simple Storage Service 使用者指南，包括支援的演算法。

您可以靈活地選擇最符合您需求的演算法，並讓 SDK 計算檢查總和。或者，您可以使用其中一個支援的演算法來提供預先計算的檢查總和值。

> **Note**
>
> 從 1.4.0 版開始 適用於 Kotlin 的 AWS SDK，會自動計算上傳的CRC32檢查總和，以提供預設完整性保護。如果您未提供預先計算的檢查總和值，或者您未指定 SDK 應該用來計算檢查總和的演算法，則 SDK 會計算此檢查總和。
> 軟體開發套件還提供全域設定，用於外部設定的資料完整性保護，您可以在軟體AWS SDKs和工具參考指南中閱讀這些保護。

我們討論兩個請求階段的檢查總和：上傳物件和下載物件。

上傳物件

您可以使用 `putObject`函數搭配請求參數，搭配適用於 Kotlin 的 SDK 將物件上傳至 Amazon S3。請求資料類型提供 `checksumAlgorithm` 屬性以啟用檢查總和運算。

下列程式碼片段顯示使用CRC32檢查總和上傳物件的請求。當 SDK 傳送請求時，它會計算CRC32檢查總和並上傳物件。Amazon S3 會將檢查總和與 物件一起存放。

```
val request = PutObjectRequest {
```

```
    bucket = "amzn-s3-demo-bucket"
    key = "key"
    checksumAlgorithm = ChecksumAlgorithm.CRC32
}
```

如果您未隨請求提供檢查總和演算法，檢查總和行為會根據您使用的 SDK 版本而有所不同，如下表所示。

未提供檢查總和演算法時的檢查總和行為

Kotlin SDK 版本	檢查總和行為
早於 1.4.0	軟體開發套件不會自動計算以 CRC 為基礎的檢查總和，並在請求中提供它。
1.4.0 或更新版本	SDK 使用CRC32演算法來計算檢查總和，並在請求中提供檢查總和。Amazon S3 透過計算自己的CRC32檢查總和來驗證傳輸的完整性，並將其與 SDK 提供的檢查總和進行比較。如果檢查總和相符，檢查總和會與 物件一起儲存。

使用預先計算的檢查總和值

隨請求提供的預先計算檢查總和值會停用 SDK 的自動運算，並改用提供的值。

下列範例顯示具有預先計算 SHA256 檢查總和的請求。

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

如果 Amazon S3 判斷指定演算法的檢查總和值不正確，則服務會傳回錯誤回應。

分段上傳

您也可以搭配分段上傳使用檢查總和。

您必須在CreateMultipartUpload請求和每個UploadPart請求中指定檢查總和演算法。最後，您必須在 中指定每個部分的檢查總和CompleteMultipartUpload。下列範例說明如何使用指定的檢查總和演算法建立分段上傳。

```
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
    }
}
```

下載物件

當您使用 getObject getObject方法下載物件時，開發套件會在當 的建置器 checksumMode 屬性GetObjectRequest設定為 時ChecksumMode.Enabled。

以下程式碼片段中的請求會指示 SDK 透過計算檢查總和並比較值來驗證回應中的檢查總和。

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

> **Note**
>
> 如果物件未使用檢查總和上傳，則不會進行驗證。

如果您使用 SDK 版本 1.4.0 或更新版本，軟體開發套件會自動檢查`getObject`請求的完整性，而不會`checksumMode = ChecksumMode.Enabled`新增至請求。

非同步驗證

由於適用於 Kotlin 的 SDK 會在從 Amazon S3 下載物件時使用串流回應，因此檢查總和將在您取用物件時計算。因此，您必須使用 物件，才能驗證檢查總和。

下列範例顯示如何完全使用回應來驗證檢查總和。

```
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.
}
```

相反地，以下範例中的程式碼不會以任何方式使用 物件，因此檢查總和不會進行驗證。

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

如果 SDK 計算的檢查總和不符合與回應一起傳送的預期檢查總和，則 SDK 會擲回 `ChecksumMismatchException`。

使用適用於 Kotlin 的 SDK 來使用 Amazon S3 多區域存取點

Amazon S3 多區域存取點提供全域端點，應用程式可用來滿足來自位於多個 Amazon S3 儲存貯體的請求 AWS 區域。您可以使用多區域存取點，以單一區域中使用的相同架構建置多區域應用程式，然後在世界任何地方執行這些應用程式。

Amazon S3 使用者指南包含有關多區域存取點的更多背景資訊。

使用多區域存取點

若要建立多區域存取點，請先在您要提供請求的每個 AWS 區域中指定一個儲存貯體。下列程式碼片段會建立兩個儲存貯體。

建立儲存貯體

下列 函數會建立兩個儲存貯體，以使用多區域存取點。一個儲存貯體位於 區域`us-east-1`，另一個儲存貯體位於 區域`us-west-1`。

做為第一個引數傳入的 S3 用戶端的建立會顯示在 下的第一個範例中the section called “使用 物件”。

```
        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.")
            }
        }
```

您可以使用 Kotlin SDK 的 S3 控制用戶端來建立、刪除和取得多區域存取點的相關資訊。

在 S3 控制成品上新增相依性，如下列程式碼片段所示。（您可以導覽至 *X.Y.Z* 連結，以查看可用的最新版本。）

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

將 S3 控制用戶端設定為使用 AWS 區域 `us-west-2` ，如下列程式碼所示。所有 S3 控制用戶端操作都必須以`us-west-2`區域為目標。

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

使用 S3 控制用戶端透過指定儲存貯體名稱（先前建立）來建立多區域存取點，如下列程式碼所示。

```
    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"
    }
```

由於多區域存取點的建立是非同步操作，因此您可以使用從立即回應中收到的字符來檢查建立程序的狀態。狀態檢查傳回成功訊息後，您可以使用 GetMultiRegionAccessPoint操作來取得多區域存取點的別名。別名是 ARN 的最後一個元件，您需要此元件才能進行物件層級操作。

使用字符檢查狀態

使用 DescribeMultiRegionAccessPointOperation檢查上次操作的狀態。
在requestStatus值變成「SUCCEEDED」之後，您就可以使用多區域存取點。

```
        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)
            }
        }
```

使用物件和多區域存取點

您可以使用 S3 用戶端來處理多區域存取點中的物件。您可以在多區域存取點上使用的儲存貯體中物件上的許多操作。如需詳細資訊和完整的操作清單，請參閱多區域存取點與 S3 操作的相容性。

使用多區域存取點的操作會使用非對稱 SigV4 (SigV4a) 簽署演算法進行簽署。若要設定 SigV4a，請先將下列相依性新增至您的專案。（您可以導覽至 *X.Y.Z* 連結，以查看可用的最新版本。）

```
...
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")
...
```

新增相依性之後，請將 S3 用戶端設定為使用 SigV4a 簽署演算法，如下列程式碼所示。

```
        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
        }
```

設定 S3 用戶端之後，S3 針對多區域存取點支援的操作也會運作相同。唯一的差別是儲存貯體參數必須是多區域存取點的 ARN。您可以從 Amazon S3 主控台或以程式設計方式取得 ARN，如先前傳回 ARN 的 `createMrap`函數所示。

下列程式碼範例顯示 `GetObject`操作中使用的 ARN。

```
    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
    }
```

使用 使用 DynamoDB 適用於 Kotlin 的 AWS SDK

使用以 AWS 帳戶為基礎的端點

DynamoDB 提供以AWS 帳戶為基礎的端點，可透過使用 AWS 您的帳戶 ID 來簡化請求路由來改善效能。

若要利用此功能，您需要使用 1.3.37 版或更新版本的 適用於 Kotlin 的 AWS SDK。您可以在 Maven 中央儲存庫中找到最新版本的 SDK。支援版本的 SDK 處於作用中狀態後，它會自動使用新的端點。

如果您想要選擇退出以帳戶為基礎的路由，您有四個選項：

- 將 DynamoDB 服務用戶端`AccountIdEndpointMode`設定為 `DISABLED`。
- 設定環境變數。
- 設定 JVM 系統屬性。
- 更新共用 AWS 組態檔案設定。

下列程式碼片段示範如何透過設定 DynamoDB 服務用戶端來停用帳戶型路由：

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

AWS SDKs 和工具參考指南提供有關最後三個組態選項的詳細資訊。

使用 DynamoDB Mapper 將類別映射至 DynamoDB 項目 （開發人員預覽版）

⚠ DynamoDB Mapper 是開發人員預覽版本。其功能不完整，可能會有所變更。

DynamoDB Mapper 是一個高階程式庫，提供將 Kotlin 類別映射至 DynamoDB 資料表和索引的機制，類似於 AWS SDK for Java的 DynamoDB 增強型用戶端或 AWS SDK for .NET的物件持久性模型。

您可以定義描述資料物件的結構描述，以及如何將其轉換為 DynamoDB 項目。定義結構描述後，DynamoDB Mapper 會提供直覺式界面，在資料表和索引的建立、讀取、更新或刪除 (CRUD) 操作中使用您的物件。

主題

- DynamoDB Mapper 入門
- 設定 DynamoDB Mapper
- 從註釋產生結構描述
- 手動定義結構描述
- 搭配 DynamoDB Mapper 使用次要索引
- 使用表達式

DynamoDB Mapper 入門

⚠ DynamoDB Mapper 是開發人員預覽版本。其功能不完整，可能會有所變更。

下列教學課程介紹 DynamoDB Mapper 的基本元件，並說明如何在程式碼中使用它。

新增相依性

若要開始在 Gradle 專案中使用 DynamoDB Mapper，請將外掛程式和兩個相依性新增至 `build.gradle.kts` 檔案。

（您可以導覽至 *X.Y.Z* 連結，以查看可用的最新版本。）

```
// 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")
}
```

*將 *<Version>* 取代為最新版本的 SDK。若要尋找最新版本的 SDK，請檢查 GitHub 上的最新版本。

> **Note**
>
> 如果您打算手動定義結構描述，其中一些相依性是選用的。the section called “手動定義結構描述” 如需詳細資訊和減少的相依性集，請參閱 。

建立和使用映射器

DynamoDB Mapper 使用 適用於 Kotlin 的 AWS SDK的 DynamoDB 用戶端與 DynamoDB 互動。當您建立映射器`DynamoDbClient`執行個體時，您需要提供完全設定的執行個體，如下列程式碼片段所示：

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

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

> **Note**
>
> `DynamoDbMapper` 不支援資料表建立操作。使用 `DynamoDbClient`建立資料表。

建立映射器執行個體之後，您可以使用它來取得資料表執行個體，如下所示：

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

先前的程式碼會使用 定義的結構描述，取得 中DynamoDB名為 cars 之資料表的參考 CarSchema（我們將在下面討論結構描述）。建立資料表執行個體之後，您可以對其執行操作。下列程式碼片段顯示cars資料表的兩個範例操作：

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

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

先前的程式碼會在cars資料表中建立新的項目。此程式碼會使用 Car類別內嵌建立Car執行個體，其定義如下所示。接下來，程式碼會查詢cars資料表中分割區索引鍵為 的項目，Peugeot並列印它們。操作的詳細說明如下。

使用類別註釋定義結構描述

對於各種 Kotlin 類別，開發套件可以使用適用於 Gradle 的 DynamoDB Mapper 結構描述產生器外掛程式，在建置時間自動產生結構描述。當您使用結構描述產生器時，軟體開發套件會檢查您的類別來推斷結構描述，這會緩解手動定義結構描述時涉及的一些樣板。您可以使用其他註釋和組態來自訂產生的結構描述。

若要從註釋產生結構描述，請先使用 註釋您的類別，並使用 @DynamoDbItem 和 @DynamoDbPartitionKey 註釋任何索引鍵@DynamoDbSortKey。下列程式碼顯示註釋的Car類別：

```
// 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
)
```

建置之後，您可以參考自動產生的 CarSchema。您可以使用映射器getTable方法中的 參考來取得資料表執行個體，如下所示：

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

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

或者，您可以利用建置時DynamoDbMapper自動產生的 延伸方法來取得資料表執行個體。透過使用此方法，您不需要依名稱參考結構描述。如下所示，自動產生的getCarsTable延伸方法會傳回資料表執行個體的參考：

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

如需詳細資訊和範例，請參閱 the section called “產生結構描述”。

叫用 操作

DynamoDB Mapper 支援開發套件 上可用操作的子集DynamoDbClient。Mapper 操作的名稱與 SDK 用戶端上的對應操作相同。許多映射器請求/回應成員與其 SDK 用戶端相同，但有些已重新命名、重新輸入或捨棄。

您可以使用 DSL 語法在資料表執行個體上叫用 操作，如下所示：

```
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)
```

您也可以使用明確請求物件叫用 操作：

```
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)
```

前兩個程式碼範例是相等的。

使用分頁回應

有些操作，例如 query和 scan可以傳回可能太大而無法在單一回應中傳回的資料收集。為了確保處理所有物件，DynamoDB Mapper 提供分頁方法，不會立即呼叫 DynamoDB，而是傳回Flow操作回應類型的 ，如下所示Flow<ScanResponse<Car>>：

```
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) }
}
```

通常，物件流程對於商業邏輯比包含物件的回應流程更有用。映射器提供分頁回應的延伸方法，以存取物件的流程。例如，下列程式碼會傳回 Flow<Car>而非 Flow<ScanResponse<Car>>，如先前所示：

```
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) }
```

設定 DynamoDB Mapper

⚠ DynamoDB Mapper 是開發人員預覽版本。其功能不完整，可能會有所變更。

DynamoDB Mapper 提供組態選項，您可以使用自訂程式庫的行為以符合您的應用程式。

使用攔截器

DynamoDB Mapper 程式庫定義您可以在映射器請求管道的關鍵階段使用掛鉤。您可以實作`Interceptor`界面來實作勾點，以觀察或修改映射器程序。

您可以將單一 DynamoDB Mapper 上的一或多個攔截器註冊為組態選項。如需如何註冊攔截器，請參閱本節結尾的範例。

了解請求管道

映射器的請求管道包含下列 5 個步驟：

1. 初始化：設定 操作並收集初始內容。
2. 序列化：將高階請求物件轉換為低階請求物件。此步驟會將高階 Kotlin 物件轉換為由屬性名稱和值組成的 DynamoDB 項目。
3. 低階調用：在基礎 DynamoDB 用戶端上執行請求。
4. 去序列化：將低階回應物件轉換為高階回應物件。此步驟包含將包含屬性名稱和值的 DynamoDB 項目轉換為高階 Kotlin 物件。
5. 完成：完成高階回應以傳回給發起人。如果在管道執行期間擲回例外狀況，此步驟會完成擲回給發起人的例外狀況。

勾點

勾點是映射器在管道中特定步驟之前或之後調用的攔截器方法。勾點有兩種變體：唯讀和修改（或讀寫）。例如，`readBeforeInvocation` 是唯讀勾點，映射器會在低階調用步驟之前在 階段中執行。

唯讀勾點

映射器會在管道中每個步驟之前和之後叫用唯讀勾點 (初始化步驟之前和完成步驟之後除外）。唯讀機罩提供正在進行的高階操作的唯讀檢視。它們提供一種機制來檢查 操作的狀態，例如記錄、偵錯、收集指標。每個唯讀勾點都會接收內容引數，並傳回 `Unit`。

映射器會擷取在唯讀勾點期間擲回的任何例外狀況，並將其新增至內容。然後，它會將具有例外狀況的內容傳遞至相同階段中的後續攔截器勾點。映射器只會在呼叫最後一個攔截器相同階段的唯讀勾點之後，才會將任何例外狀況擲回給發起人。例如，如果映射器設定了兩個攔截器A和B，而 A的readAfterSerialization勾點擲回例外狀況，映射器會將例外狀況新增至傳遞至B勾readAfterSerialization點的內容。B的readAfterSerialization勾點完成後，映射器會將例外狀況擲回給發起人。

修改掛鉤

映射器會在管道中的每個步驟之前叫用修改掛鉤 (初始化之前除外）。修改掛鉤可讓您查看和修改正在進行的高階操作。它們可用來以映射器組態和項目結構描述未採用的方式自訂行為和資料。每個修改勾點都會收到內容引數，並因此傳回該內容的一些子集 - 由勾點修改或從輸入內容傳遞。

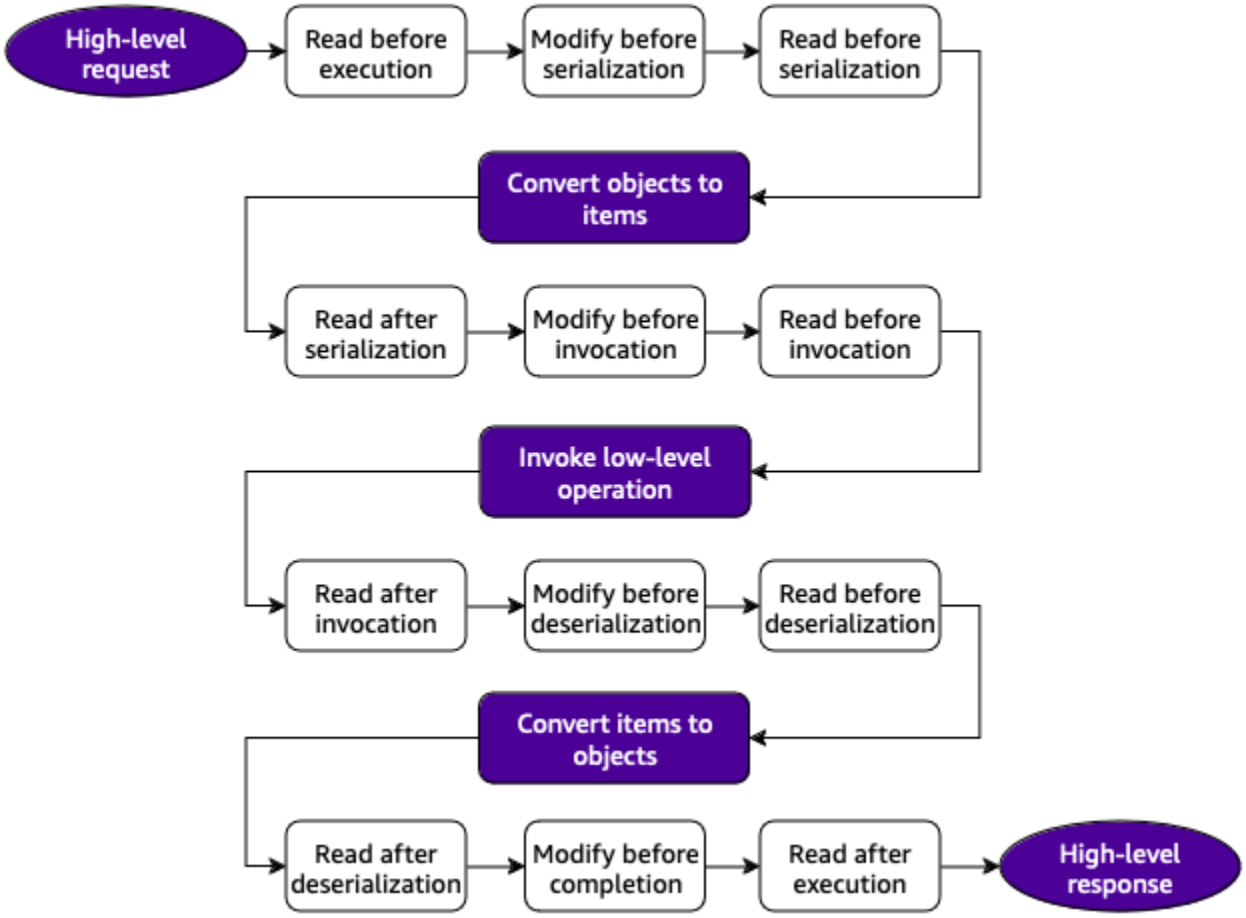
如果映射器在執行修改勾點時擷取任何例外狀況，則不會在相同階段執行任何其他攔截器的修改勾點。映射器會將例外狀況新增至內容，並將其傳遞至下一個唯讀勾點。映射器只會在呼叫最後一個攔截者的相同階段唯讀勾點之後，才會將任何例外狀況擲回給發起人。例如，如果映射器設定了兩個攔截器A和 B，而 A的modifyBeforeSerialization勾點擲回例外狀況，則不會叫用B的modifyBeforeSerialization勾點。將執行攔截器 A和 B's readAfterSerialization hook，之後會將例外狀況擲回給發起人。

執行順序

在映射器的組態中定義攔截器的順序決定映射器呼叫勾點的順序：

- 對於低階調用步驟之前的階段，它會以其在組態中新增的相同順序執行掛鉤。
- 對於低階調用步驟之後的階段，它會以與組態中新增的順序相反的順序執行勾點。

下圖顯示勾點方法的執行順序：



勾點方法執行順序的文字描述

映射器會依下列順序執行攔截器的勾點：

1. DynamoDB Mapper 調用高階請求
2. 執行前讀取
3. 在序列化之前修改
4. 序列化前讀取
5. DynamoDB Mapper 會將物件轉換為項目
6. 序列化後讀取
7. 叫用前修改
8. 叫用前讀取
9. DynamoDB Mapper 調用低階操作
10.叫用後讀取
11.在還原序列化之前修改

12還原序列化前讀取

13DynamoDB Mapper 會將項目轉換為物件

14還原序列化後讀取

15完成前修改

16執行後讀取

17DynamoDB Mapper 傳回高階回應

範例組態

下列範例示範如何在DynamoDbMapper執行個體上設定攔截器：

```
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
}
```

從註釋產生結構描述

⚠ DynamoDB Mapper 是開發人員預覽版本。其功能不完整，可能會有所變更。

DynamoDB Mapper 依賴於定義 Kotlin 類別與 DynamoDB 項目之間映射的結構描述。您的 Kotlin 類別可以使用結構描述產生器 Gradle 外掛程式來推動結構描述的建立。

套用外掛程式

若要開始為您的類別產生程式碼結構描述，請在應用程式的建置指令碼中套用外掛程式，並在註釋模組上新增相依性。下列 Gradle 指令碼程式碼片段顯示產生程式碼所需的設定。

（您可以導覽至 *X.Y.Z* 連結，以查看可用的最新版本。）

```
// 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")
}
```

設定 外掛程式

外掛程式提供許多組態選項，您可以使用建置指令碼中的dynamoDbMapper { ... }外掛程式延伸來套用：

選項	選項描述	值
generateBuilderClasses	控制是否為標註 的類別產生 DSL 樣式建置器類別 @DynamoDbItem	WHEN_REQUIRED （預設）：不會對僅包含公有可變成員且具有零位元建構函數的類別產生建置器類別 ALWAYS：一律會產生建置器類別
visibility	控制所產生類別的可見性	PUBLIC (default) INTERNAL

選項	選項描述	值
destinationPackage	指定所產生類別的套件名稱	RELATIVE（預設）：結構描述類別會在相對於註釋類別的子套件中產生。依預設，子套件名為 dynamodbmapper.generatedschemas ，可透過傳遞字串參數來設定 ABSOLUTE：結構描述類別會在相對於應用程式根目錄的絕對套件中產生。根據預設，套件名為 aws.sdk.kotlin.hll.dynamodbmapper.generatedschemas ，且可透過傳遞字串參數來設定。
generateGetTableExtension	控制是否會產生DynamoDbMapper.get${CLASS_NAME}Table 延伸方法	true (default) false

Example程式碼產生外掛程式組態的範例

下列範例會設定目的地套件和產生結構描述的可見性：

```
// 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
}
```

註釋類別

結構描述產生器會尋找類別註釋，以決定要產生結構描述的類別。若要選擇加入產生結構描述，請使用 標註您的類別@DynamoDbItem。您也必須將做為項目分割區索引鍵的類別屬性加上註釋@DynamoDbPartitionKey。

下列類別定義顯示產生結構描述所需的最低註釋：

Example

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

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

類別註釋

下列註釋會套用至類別以控制結構描述產生：

- @DynamoDbItem：指定此類別/界面描述資料表中的項目類型。除非明確忽略，否則此類型的所有公有屬性都會對應至屬性。出現時，將會為此類別產生結構描述。
  - converterName：選用參數，指出應使用自訂結構描述，而不是結構描述產生器外掛程式所建立的。這是自訂ItemConverter類別的完整名稱。the section called “定義自訂項目轉換器” 本節顯示建立和使用自訂結構描述的範例。

屬性註釋

您可以將下列註釋套用至類別屬性，以控制結構描述產生：

- @DynamoDbPartitionKey：指定項目的分割區索引鍵。
- @DynamoDbSortKey：指定項目的選用排序索引鍵。
- @DynamoDbIgnore：指定此類別屬性不應由 DynamoDB Mapper 轉換為項目屬性或從中轉換。
- @DynamoDbAttribute：指定此類別屬性的選用自訂屬性名稱。

定義自訂項目轉換器

在某些情況下，您可能想要為 類別定義自訂項目轉換器。其中一個原因是，如果您的類別使用結構描述產生器外掛程式不支援的類型。我們使用下列版本的 Employee類別做為範例：

```
import kotlin.uuid.Uuid

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

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

Employee 類別現在使用 類型，結構描述產生器目前不支援此kotlin.uuid.Uuid類型。產生結構描述失敗並發生錯誤：Unsupported attribute type TypeRef(pkg=kotlin.uuid, shortName=Uuid, genericArgs=[], nullable=false)。此錯誤表示外掛程式無法為此類別產生項目轉換器。因此，我們需要撰寫自己的 。

為此，我們為 類別實作 ItemConverter ，然後透過指定新項目轉換器的完整名稱來修改@DynamoDbItem類別註釋。

首先，我們為 kotlin.uuid.Uuid類別實作 ValueConverter ：

```
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())
}
```

然後，我們為 Employee類別實作 ItemConverter 。會在 "workstationId" 的屬性描述項中ItemConverter使用此新值轉換器：

```
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
        )
    ),
)
```

現在我們已經定義項目轉換器，我們可以將其套用至我們的類別。我們會提供完整類別名稱來更新@DynamoDbItem註釋以參考項目轉換器，如下所示：

```
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，我們可以開始將 類別與 DynamoDB Mapper 搭配使用。

手動定義結構描述

⚠ DynamoDB Mapper 是開發人員預覽版本。其功能不完整，可能會有所變更。

在程式碼中定義結構描述

為了獲得最大的控制和可自訂性，您可以在程式碼中手動定義和自訂結構描述。

如下列程式碼片段所示，相較於使用註釋驅動的結構描述建立，您需要在 `build.gradle.kts` 檔案中包含較少的相依性。

（您可以導覽至 *X.Y.Z* 連結，以查看可用的最新版本。）

```
// 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.
}
```

請注意，您不需要結構描述產生器外掛程式或註釋套件。

Kotlin 類別和 DynamoDB 項目之間的映射需要`ItemSchema<T>`實作，其中 T是 Kotlin 類別的類型。結構描述包含下列元素：

- 項目轉換器，定義如何在 Kotlin 物件執行個體和 DynamoDB 項目之間轉換。
- 分割區索引鍵規格，定義分割區索引鍵屬性的名稱和類型。
- 或者，排序索引鍵規格，定義排序索引鍵屬性的名稱和類型。

在下列程式碼中，我們會手動建立CarSchema執行個體：

```
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)
```

先前的程式碼會建立名為 的轉換器carConverter，其定義為 的匿名實作ItemConverter<Car>。轉換器的 convertTo方法接受Car引數，並傳回代表 DynamoDB 項目屬性常值索引鍵和值的Item執行個體。轉換器的 convertFrom方法接受 Item引數，並從Item引數的屬性值傳回Car執行個體。

接下來，程式碼會建立兩個索引鍵規格：一個用於分割區索引鍵，另一個用於排序索引鍵。每個 DynamoDB 資料表或索引都必須只有一個分割區索引鍵，因此每個 DynamoDB Mapper 結構描述定義都必須相應。結構描述也可能有一個排序索引鍵。

在最後一個陳述式中，程式碼會從轉換器和金鑰規格建立 cars DynamoDB 資料表的結構描述。

產生的結構描述等同於我們在 the section called “使用類別註釋定義結構描述”區段中產生的註釋驅動結構描述。以下是我們使用的註釋類別，以供參考：

具有 DynamoDB Mapper 註釋的汽車類別

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

    @DynamoDbSortKey
    val model: String,

    val initialYear: Int
)
```

除了實作您自己的 之外`ItemConverter`，DynamoDB Mapper 還包含數個實用的實作，例如：

- `SimpleItemConverter`：使用建置器類別和屬性描述項提供簡單的轉換邏輯。如需the section called “定義自訂項目轉換器”如何使用此實作，請參閱 中的範例。
- `HeterogeneousItemConverter`：透過使用辨別器屬性和委派子類型的`ItemConverter`執行個體，提供多態類型轉換邏輯。
- `DocumentConverter`：為 `Document` 物件中的非結構化資料提供轉換邏輯。

搭配 DynamoDB Mapper 使用次要索引

⚠ DynamoDB Mapper 是開發人員預覽版本。其功能不完整，可能會有所變更。

定義次要索引的結構描述

DynamoDB 資料表支援次要索引，該索引可使用與基礎資料表本身上定義的索引鍵不同的索引鍵來存取資料。如同基礎資料表，DynamoDB Mapper 會使用 `ItemSchema`類型與索引互動。

DynamoDB 次要索引不需要包含基礎資料表中的每個屬性。因此，對應至索引的 Kotlin 類別可能與對應至該索引基礎資料表的 Kotlin 類別不同。在這種情況下，必須為索引類別宣告單獨的結構描述。

下列程式碼會手動建立 DynamoDB `cars`資料表的索引結構描述。

```
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
        )
*/
```

我們現在可以在 操作中使用Model執行個體。

在 操作中使用次要索引

DynamoDB Mapper 支援索引上的操作子集，即 queryPaginated和 scanPaginated。若要在索引上調用這些操作，您必須先從資料表物件取得索引的參考。在下列範例中，我們使用先前為cars-by-model索引modelSchema建立的 （此處未顯示建立）：

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

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

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

使用表達式

⚠ DynamoDB Mapper 是開發人員預覽版本。其功能不完整，可能會有所變更。

某些 DynamoDB 操作接受您可以用來指定限制條件或條件的表達式。DynamoDB Mapper 提供慣用的 Kotlin DSL 來建立表達式。DSL 為您的程式碼提供更大的結構和可讀性，也可讓您更輕鬆地撰寫表達式。

本節說明 DSL 語法並提供各種範例。

在 操作中使用表達式

您可以在 等操作中使用表達式scan，它們會根據您定義的條件篩選傳回的項目。若要搭配 DynamoDB Mapper 使用表達式，請在操作請求中新增表達式元件。

下列程式碼片段顯示 scan操作中使用的篩選條件表達式範例。它使用 lambda 引數來描述篩選條件，限制項目傳回至year屬性值為 2001 的項目：

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

table.scanPaginated {
```

```
    filter {
        attr("year") eq 2001
    }
}
```

下列範例顯示支援兩個位置表達式query的操作：排序索引鍵篩選和非索引鍵篩選：

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

先前的程式碼會篩選符合所有三個條件的結果：

- 分割區索引鍵屬性值為 1000 -AND-
- 排序索引鍵屬性值以字母 M -AND- 開頭
- year 屬性值為 2001

DSL 元件

DSL 語法會公開您用來建置表達式的多種元件類型，如下所述。

屬性

大多數條件參考屬性，由其金鑰或文件路徑識別。使用 DSK，您可以使用 attr函數建立所有屬性參考，並選擇性地進行其他修改。

下列程式碼顯示從簡單到複雜的範例屬性參考範圍，例如索引的清單取消參考和索引鍵的映射取消參考：

```
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`.
```

等式和不等式

您可以依等式和不等式比較表達式中的屬性值。您可以將屬性值與常值或其他屬性值進行比較。您用來指定條件的函數包括：

- eq： 等於 （ 相當於 ==)
- neq： 不等於 （ 相當於 !=)
- gt： 大於 （ 相當於 >)
- gte：大於或等於 （ 相當於 >=)
- lt： 小於 （ 相當於 <)
- lte：小於或等於 （ 相當於 <=)

您可以使用 infix 標記法將比較函數與引數結合，如下列範例所示：

```
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`.
```

範圍和集

除了單一值之外，您還可以將屬性值與範圍或集合中的多個值進行比較。您可以使用 infix `isIn`函數進行比較，如下列範例所示：

```
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",
)
```

isIn 函數為集合（例如 Set<String>) 和您可以表達為 Kotlin 的邊界 ClosedRange<T>（例如 IntRange) 提供過載。對於您無法表達為 的邊界 ClosedRange<T>（例如位元組陣列或其他屬性參考），您可以使用 isBetween函數：

```
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`.
```

布林值邏輯

您可以使用下列函數結合個別條件或使用布林邏輯進行修改：

- and：每個條件都必須是 true（等於 &&)
- or：至少一個條件必須為 true（相當於 ||)
- not：指定的條件必須是 false（相當於 !)

下列範例顯示每個 函數：

```
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.
)
```

您可以依布林函數進一步合併布林值條件，以建立巢狀邏輯，如下列表達式所示：

```
or(
    and(
        attr("foo") eq 123,
        attr("bar") eq "abc",
    ),
    and(
        attr("foo") eq 234,
        attr("bar") eq "bcd",
    ),
)
```

先前的表達式會篩選符合下列任一條件的結果：

- 這兩個條件都是 true：
  - foo 屬性值為 123 -AND-
  - bar 屬性值為 "abc"
- 這兩個條件都是 true：
  - foo 屬性值為 234 -AND-
  - bar 屬性值為 "bcd"

這相當於下列 Kotlin 布林表達式：

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

函數和屬性

下列函數和屬性提供額外的表達式功能：

- contains：檢查字串/清單屬性值是否包含指定的值
- exists：檢查是否已定義屬性並保留任何值（包括 null）
- notExists：檢查屬性是否未定義
- isOfType：檢查屬性值是否為指定類型，例如字串、數字、布林值等
- size：取得屬性的大小，例如集合中的元素數量或字串的長度
- startsWith：檢查字串屬性值是否以指定的子字串開頭

下列範例顯示您可以在表達式中使用的其他函數和屬性的使用：

```
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.
```

排序金鑰篩選條件

排序索引鍵（例如在query操作的 keyCondition 參數中）上的篩選條件表達式不會使用具名屬性值。若要在篩選條件中使用排序索引鍵，您必須在所有比較sortKey中使用 關鍵字。sortKey 關鍵字會取代 attr("<sort key name>")，如下列範例所示：

```
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).
```

您不能將排序索引鍵篩選條件與布林邏輯結合，它們僅支援上述比較的子集：

- 相等性和不相等性：支援的所有比較
- 範圍和集：支援的所有比較
- 布林值邏輯：不支援
- 函數和屬性：僅startsWith支援

適用於 Kotlin 的 SDK 程式碼範例

本主題中的程式碼範例示範如何使用適用於 Kotlin 的AWS SDK AWS。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

有些服務包含其他範例類別，示範如何利用特定於服務的程式庫或函數。

服務

- 使用適用於 Kotlin 的 SDK 的 API Gateway 範例
- 使用適用於 Kotlin 的 SDK 的 Aurora 範例
- 使用適用於 Kotlin 的 SDK 的 Auto Scaling 範例
- 使用適用於 Kotlin 的 SDK 的 Amazon Bedrock 範例
- 使用適用於 Kotlin 的 SDK 的 Amazon Bedrock 執行時期範例
- 使用適用於 Kotlin 的 SDK 的 CloudWatch 範例
- 使用適用於 Kotlin 的 SDK 的 CloudWatch Logs 範例
- 使用適用於 Kotlin 的 SDK 的 Amazon Cognito 身分提供者範例
- 使用適用於 Kotlin 的 SDK 的 Amazon Comprehend 範例
- 使用適用於 Kotlin 的 SDK 的 DynamoDB 範例
- 使用適用於 Kotlin 的 SDK 的 Amazon EC2 範例
- 使用適用於 Kotlin 的 SDK 的 Amazon ECR 範例
- 使用適用於 Kotlin 的 SDK 的 OpenSearch Service 範例
- 使用適用於 Kotlin 的 SDK 的 EventBridge 範例
- AWS Glue使用適用於 Kotlin 的 SDK 的範例
- 使用適用於 Kotlin 的 SDK 的 IAM 範例
- AWS IoT使用適用於 Kotlin 的 SDK 的範例
- AWS IoT data使用適用於 Kotlin 的 SDK 的範例

- 使用適用於 Kotlin 的 SDK 的 Amazon Keyspaces 範例
- AWS KMS使用適用於 Kotlin 的 SDK 的範例
- 使用適用於 Kotlin 的 SDK 的 Lambda 範例
- 使用適用於 Kotlin 的 SDK 的 Amazon Location 範例
- 使用適用於 Kotlin 的 SDK 的 MediaConvert 範例
- 使用適用於 Kotlin 的 SDK 的 Amazon Pinpoint 範例
- 使用適用於 Kotlin 的 SDK 的 Amazon RDS 範例
- 使用適用於 Kotlin 的 SDK 的 Amazon RDS Data Service 範例
- 使用適用於 Kotlin 的 SDK 的 Amazon Redshift 範例
- 使用適用於 Kotlin 的 SDK 的 Amazon Rekognition 範例
- 使用適用於 Kotlin 的 SDK 的 Route 53 網域註冊範例
- 使用適用於 Kotlin 的 SDK 的 Amazon S3 範例
- 使用適用於 Kotlin 的 SDK 的 SageMaker AI 範例
- 使用適用於 Kotlin 的 SDK 的 Secrets Manager 範例
- 使用適用於 Kotlin 的 SDK 的 Amazon SES 範例
- 使用適用於 Kotlin 的 SDK 的 Amazon SNS 範例
- 使用適用於 Kotlin 的 SDK 的 Amazon SQS 範例
- 使用適用於 Kotlin 的 SDK 的 Step Functions 範例
- 支援使用適用於 Kotlin 的 SDK 的範例
- 使用適用於 Kotlin 的 SDK 的 Amazon Translate 範例
- 使用適用於 Kotlin 的 SDK 的 X-Ray 範例

使用適用於 Kotlin 的 SDK 的 API Gateway 範例

下列程式碼範例示範如何使用AWS SDK for Kotlin 搭配 API Gateway 來執行動作和實作常見案例。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 案例

案例

建立無伺服器應用程式來管理相片

下列程式碼範例示範如何建立無伺服器應用程式，讓使用者以標籤管理相片。

SDK for Kotlin

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

使用適用於 Kotlin 的 SDK 的 Aurora 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Aurora 來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 基本概念
- 動作

- 案例

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立自訂 Aurora 資料庫叢集參數群組並設定參數值。
- 建立使用該參數群組的資料庫叢集。
- 建立包含該資料庫的資料庫執行個體。
- 拍攝該資料庫叢集的快照，並清理資源。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

動作

`CreateDBCluster`

以下程式碼範例顯示如何使用 `CreateDBCluster`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateDBCluster。

`CreateDBClusterParameterGroup`

以下程式碼範例顯示如何使用 `CreateDBClusterParameterGroup`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateDBClusterParameterGroup。

`CreateDBClusterSnapshot`

以下程式碼範例顯示如何使用 `CreateDBClusterSnapshot`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateDBClusterSnapshot。

`CreateDBInstance`

以下程式碼範例顯示如何使用 `CreateDBInstance`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
```

```
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateDBInstance。

`DeleteDBCluster`

以下程式碼範例顯示如何使用 `DeleteDBCluster`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteDBCluster。

DeleteDBClusterParameterGroup

以下程式碼範例顯示如何使用 DeleteDBClusterParameterGroup。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { 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.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteDBClusterParameterGroup。

`DeleteDBInstance`

以下程式碼範例顯示如何使用 `DeleteDBInstance`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteDBInstance。

`DescribeDBClusterParameterGroups`

以下程式碼範例顯示如何使用 `DescribeDBClusterParameterGroups`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeDBClusterParameterGroups。

`DescribeDBClusterParameters`

以下程式碼範例顯示如何使用 `DescribeDBClusterParameters`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { 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}")
                }
            }
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeDBClusterParameters。

`DescribeDBClusterSnapshots`

以下程式碼範例顯示如何使用 `DescribeDBClusterSnapshots`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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!")
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeDBClusterSnapshots。

`DescribeDBClusters`

以下程式碼範例顯示如何使用 `DescribeDBClusters`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
```

```
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { 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}")
                }
            }
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeDBClusters。

`DescribeDBEngineVersions`

以下程式碼範例顯示如何使用 `DescribeDBEngineVersions`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeDBEngineVersions。

`DescribeDBInstances`

以下程式碼範例顯示如何使用 `DescribeDBInstances`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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")
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeDBInstances。

ModifyDBClusterParameterGroup

以下程式碼範例顯示如何使用 ModifyDBClusterParameterGroup。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
// 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.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ModifyDBClusterParameterGroup。

案例

建立 Aurora 無伺服器工作項目追蹤器

下列程式碼示範如何建立一個 Web 應用程式，該應用程式會追蹤 Amazon Aurora Serverless 資料庫中的工作項目，並使用 Amazon Simple Email Service (Amazon SES) 傳送報告。

SDK for Kotlin

說明如何建立可追蹤和報告存放在 Amazon RDS 資料庫中的工作項目的 Web 應用程式。

如需完整的原始碼以及如何設定 Spring REST API 以查詢 Amazon Aurora 無伺服器資料並供 React 應用程式使用的說明，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Aurora
- Amazon RDS
- Amazon RDS 資料服務
- Amazon SES

使用適用於 Kotlin 的 SDK 的 Auto Scaling 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Auto Scaling 來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 基本概念
- 動作

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 以啟動範本和可用區域建立 Amazon EC2 Auto Scaling 群組，並取得有關執行中執行個體的相關資訊。
- 啟用 Amazon CloudWatch 指標收集。
- 更新群組所需的容量，並等待執行個體啟動。
- 終止群組中的執行個體。
- 列出為因應使用者請求和容量變更而發生的擴展活動。
- 取得 CloudWatch 指標的統計資料，然後清除資源。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection

- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

動作

`CreateAutoScalingGroup`

以下程式碼範例顯示如何使用 `CreateAutoScalingGroup`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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!")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateAutoScalingGroup。

`DeleteAutoScalingGroup`

以下程式碼範例顯示如何使用 `DeleteAutoScalingGroup`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DeleteAutoScalingGroup。

`DescribeAutoScalingGroups`

以下程式碼範例顯示如何使用 `DescribeAutoScalingGroups`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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)
            }
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DescribeAutoScalingGroups。

DescribeAutoScalingInstances

以下程式碼範例顯示如何使用 DescribeAutoScalingInstances。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DescribeAutoScalingInstances。

DescribeScalingActivities

以下程式碼範例顯示如何使用 DescribeScalingActivities。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DescribeScalingActivities。

`DisableMetricsCollection`

以下程式碼範例顯示如何使用 `DisableMetricsCollection`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DisableMetricsCollection。

`EnableMetricsCollection`

以下程式碼範例顯示如何使用 `EnableMetricsCollection`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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")
    }
```

```
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 EnableMetricsCollection。

`SetDesiredCapacity`

以下程式碼範例顯示如何使用 `SetDesiredCapacity`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 SetDesiredCapacity。

`TerminateInstanceInAutoScalingGroup`

以下程式碼範例顯示如何使用 `TerminateInstanceInAutoScalingGroup`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 TerminateInstanceInAutoScalingGroup。

`UpdateAutoScalingGroup`

以下程式碼範例顯示如何使用 `UpdateAutoScalingGroup`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 UpdateAutoScalingGroup。

使用適用於 Kotlin 的 SDK 的 Amazon Bedrock 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon Bedrock 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作

動作

`ListFoundationModels`

以下程式碼範例顯示如何使用 `ListFoundationModels`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

列出可用的 Amazon Bedrock 基礎模型。

```
suspend fun listFoundationModels(): List<FoundationModelSummary>? {
    BedrockClient.fromEnvironment { 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
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListFoundationModels。

使用適用於 Kotlin 的 SDK 的 Amazon Bedrock 執行時期範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon Bedrock 執行期來執行動作和實作常見案例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- Amazon Nova

Amazon Nova

Converse

下列程式碼範例示範如何使用 Bedrock 的 Converse API，將文字訊息傳送至 Amazon Nova。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

使用 Bedrock 的 Converse API，將文字訊息傳送至 Amazon Nova。

```
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)
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 Converse。

ConverseStream

下列程式碼範例示範如何使用 Bedrock 的 Converse API 將文字訊息傳送至 Amazon Nova，並即時處理回應串流。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

使用 Bedrock 的 Converse API 將文字訊息傳送至 Amazon Nova，並即時處理回應串流。

```
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()
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ConverseStream。

使用適用於 Kotlin 的 SDK 的 CloudWatch 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 CloudWatch 來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 基本概念
- 動作

開始使用

Hello CloudWatch

下列程式碼範例示範如何開始使用 CloudWatch。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
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.fromEnvironment { 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}")
            }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListMetrics。

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 列出 CloudWatch 命名空間和指標。
- 取得指標和預估帳單的統計資料。
- 建立並更新儀表板。
- 建立資料並將其新增至指標。
- 建立並觸發警示，然後檢視警示歷史記錄。
- 新增異常偵測器。
- 取得指標映像，然後清除資源。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

執行展示 CloudWatch 功能的互動式情境。

```
/**
 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector
  - PutDashboard
  - PutMetricAlarm

- PutMetricData

動作

`DeleteAlarms`

以下程式碼範例顯示如何使用 `DeleteAlarms`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteAlarm(alarmNameVal: String) {
    val request =
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}
```

- 如需 API 詳細資訊，請參閱適用於 Kotlin 的AWS SDK API 參考中的 DeleteAlarms。

`DeleteAnomalyDetector`

以下程式碼範例顯示如何使用 `DeleteAnomalyDetector`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteAnomalyDetector。

`DeleteDashboards`

以下程式碼範例顯示如何使用 `DeleteDashboards`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteDashboards。

`DescribeAlarmHistory`

以下程式碼範例顯示如何使用 `DescribeAlarmHistory`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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}")
                }
            }
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeAlarmHistory。

`DescribeAlarms`

以下程式碼範例顯示如何使用 `DescribeAlarms`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
            alarmTypes = typeList
            maxRecords = 10
        }

    CloudWatchClient.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeAlarms。

`DescribeAlarmsForMetric`

以下程式碼範例顯示如何使用 `DescribeAlarmsForMetric`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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.")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱適用於 Kotlin 的AWS SDK API 參考中的 DescribeAlarmsForMetric。

DescribeAnomalyDetectors

以下程式碼範例顯示如何使用 DescribeAnomalyDetectors。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeAnomalyDetectors。

`DisableAlarmActions`

以下程式碼範例顯示如何使用 `DisableAlarmActions`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun disableActions(alarmName: String) {
    val request =
        DisableAlarmActionsRequest {
            alarmNames = listOf(alarmName)
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.disableAlarmActions(request)
        println("Successfully disabled actions on alarm $alarmName")
    }
}
```

- 如需詳細資訊，請參閱適用於 Kotlin 的AWS SDK API 參考中的 DisableAlarmActions。

`EnableAlarmActions`

以下程式碼範例顯示如何使用 `EnableAlarmActions`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun enableActions(alarm: String) {
    val request =
        EnableAlarmActionsRequest {
            alarmNames = listOf(alarm)
        }
```

```
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.enableAlarmActions(request)
        println("Successfully enabled actions on alarm $alarm")
    }
}
```

- 如需 API 的詳細資訊，請參閱適用於 Kotlin 的AWS SDK API 參考中的 EnableAlarmActions。

`GetMetricData`

以下程式碼範例顯示如何使用 `GetMetricData`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetMetricData。

`GetMetricStatistics`

以下程式碼範例顯示如何使用 `GetMetricStatistics`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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")
            }
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetMetricStatistics。

`GetMetricWidgetImage`

以下程式碼範例顯示如何使用 `GetMetricWidgetImage`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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")
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetMetricWidgetImage。

`ListDashboards`

以下程式碼範例顯示如何使用 `ListDashboards`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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}")
```

```
            }
        }
    }
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListDashboards。

`ListMetrics`

以下程式碼範例顯示如何使用 `ListMetrics`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient.fromEnvironment { 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
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListMetrics。

`PutAnomalyDetector`

以下程式碼範例顯示如何使用 `PutAnomalyDetector`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 PutAnomalyDetector。

`PutDashboard`

以下程式碼範例顯示如何使用 `PutDashboard`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient.fromEnvironment { 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}")
                }
            }
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 PutDashboard。

PutMetricAlarm

以下程式碼範例顯示如何使用 PutMetricAlarm。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(request)
        println("Successfully created an alarm with name $alarmNameVal")
    }
```

```
}
```

- 如需 API 詳細資訊，請參閱適用於 Kotlin 的AWS SDK API 參考中的 PutMetricAlarm。

`PutMetricData`

以下程式碼範例顯示如何使用 `PutMetricData`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 PutMetricData。

使用適用於 Kotlin 的 SDK 的 CloudWatch Logs 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 CloudWatch Logs 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作

動作

`DeleteSubscriptionFilter`

以下程式碼範例顯示如何使用 `DeleteSubscriptionFilter`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteSubFilter(
    filter: String?,
    logGroup: String?,
) {
    val request =
        DeleteSubscriptionFilterRequest {
            filterName = filter
            logGroupName = logGroup
        }

    CloudWatchLogsClient.fromEnvironment { region = "us-west-2" }.use { logs ->
        logs.deleteSubscriptionFilter(request)
        println("Successfully deleted CloudWatch logs subscription filter named
 $filter")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteSubscriptionFilter。

`DescribeSubscriptionFilters`

以下程式碼範例顯示如何使用 `DescribeSubscriptionFilters`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeFilters(logGroup: String) {
    val request =
        DescribeSubscriptionFiltersRequest {
            logGroupName = logGroup
            limit = 1
        }

    CloudWatchLogsClient.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeSubscriptionFilters。

`StartLiveTail`

以下程式碼範例顯示如何使用 `StartLiveTail`。

SDK for Kotlin

包括必需的檔案。

```
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
```

開始 Live Tail 工作階段。

```
    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)
    }
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 StartLiveTail。

使用適用於 Kotlin 的 SDK 的 Amazon Cognito 身分提供者範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon Cognito 身分提供者來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作
- 案例

動作

`AdminGetUser`

以下程式碼範例顯示如何使用 `AdminGetUser`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }
```

```
    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 AdminGetUser。

`AdminInitiateAuth`

以下程式碼範例顯示如何使用 `AdminInitiateAuth`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
```

```
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 AdminInitiateAuth。

`AdminRespondToAuthChallenge`

以下程式碼範例顯示如何使用 `AdminRespondToAuthChallenge`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
// 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.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 AdminRespondToAuthChallenge。

`AssociateSoftwareToken`

以下程式碼範例顯示如何使用 `AssociateSoftwareToken`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { 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
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的 AssociateSoftwareToken。

`ConfirmSignUp`

以下程式碼範例顯示如何使用 `ConfirmSignUp`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ConfirmSignUp。

`ListUsers`

以下程式碼範例顯示如何使用 `ListUsers`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listAllUsers(userPoolId: String) {
    val request =
        ListUsersRequest {
            this.userPoolId = userPoolId
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的 ListUsers。

`ResendConfirmationCode`

以下程式碼範例顯示如何使用 `ResendConfirmationCode`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun resendConfirmationCode(
    clientIdVal: String?,
```

```
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ResendConfirmationCode。

`SignUp`

以下程式碼範例顯示如何使用 `SignUp`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 SignUp。

`VerifySoftwareToken`

以下程式碼範例顯示如何使用 `VerifySoftwareToken`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
```

```
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK 參考》中的 VerifySoftwareToken。

案例

使用需要 MFA 的使用者集區註冊使用者

以下程式碼範例顯示做法：

- 使用使用者名稱、密碼和電子郵件地址註冊並確認使用者。
- 透過將 MFA 應用程式與使用者建立關聯，以設定多重要素身分驗證。
- 使用密碼和 MFA 代碼登入。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。

- AdminGetUser
- AdminInitiateAuth
- AdminRespondToAuthChallenge
- AssociateSoftwareToken
- ConfirmDevice
- ConfirmSignUp
- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

使用適用於 Kotlin 的 SDK 的 Amazon Comprehend 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon Comprehend 來執行動作和實作常見案例。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 案例

案例

建立即時通訊軟體

下列程式碼範例示範如何搭配使用 Amazon SQS 建立即時通訊軟體。

SDK for Kotlin

示範如何使用 Amazon SQS API 開發用於傳送和擷取訊息的 Spring REST API。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Amazon Comprehend
- Amazon SQS

使用適用於 Kotlin 的 SDK 的 DynamoDB 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 DynamoDB 來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 基本概念
- 動作
- 案例

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立可存放電影資料的資料表。
- 放入、取得和更新資料表中的單個電影。
- 將影片資料從範例 JSON 檔案寫入資料表。
- 查詢特定年份發表的電影。
- 掃描某個年份範圍內發表的電影。
- 從資料表刪除電影，然後刪除資料表。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

建立 DynamoDB 資料表。

```
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.fromEnvironment { 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}")
    }
}
```

建立 Helper 函數以下載並擷取範例 JSON 檔案。

```
// 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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}
```

從資料表取得項目。

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
```

```
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

完整範例。

```
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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]}")
            }
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query

- Scan
- UpdateItem

動作

`CreateTable`

以下程式碼範例顯示如何使用 `CreateTable`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateTable。

`DeleteItem`

以下程式碼範例顯示如何使用 `DeleteItem`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteItem(request)
        println("Item with key matching $keyVal was deleted")
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteItem。

`DeleteTable`

以下程式碼範例顯示如何使用 `DeleteTable`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteDynamoDBTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteTable。

`GetItem`

以下程式碼範例顯示如何使用 `GetItem`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetItem。

`ListTables`

以下程式碼範例顯示如何使用 `ListTables`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listAllTables() {
    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.listTables(ListTablesRequest {})
        response.tableNames?.forEach { tableName ->
            println("Table name is $tableName")
        }
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListTables。

`PutItem`

以下程式碼範例顯示如何使用 `PutItem`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println(" A new item was placed into $tableNameVal.")
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 PutItem。

`Query`

以下程式碼範例顯示如何使用 `Query`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 Query。

Scan

以下程式碼範例顯示如何使用 Scan。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun scanItems(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
```

```
        }

    DynamoDbClient.fromEnvironment { 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]}")
            }
        }
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 Scan。

`UpdateItem`

以下程式碼範例顯示如何使用 `UpdateItem`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.updateItem(request)
        println("Item in $tableNameVal was updated")
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 UpdateItem。

案例

建立無伺服器應用程式來管理相片

下列程式碼範例示範如何建立無伺服器應用程式，讓使用者以標籤管理相片。

SDK for Kotlin

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3

- Amazon SNS

建立 Web 應用程式以追蹤 DynamoDB 資料

下列程式碼示範如何建立一個 Web 應用程式，該應用程式會追蹤 Amazon DynamoDB 資料表中的工作項目，並使用 Amazon Simple Email Service (Amazon SES) 傳送報告。

SDK for Kotlin

說明如何使用 Amazon DynamoDB API 來建立可追蹤 DynamoDB 工作資料的動態 Web 應用程式。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- DynamoDB
- Amazon SES

使用多批 PartiQL 陳述式查詢資料表

以下程式碼範例顯示做法：

- 透過執行多個 SELECT 陳述式取得一批項目。
- 透過執行多個 INSERT 陳述式新增一批項目。
- 透過執行多個 UPDATE 陳述式更新一批項目。
- 透過執行多個 DELETE 陳述式刪除一批項目。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun main() {
    val ddb = DynamoDbClient.fromEnvironment { 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")
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 BatchExecuteStatement。

使用 PartiQL 查詢資料表

以下程式碼範例顯示做法：

- 透過執行 SELECT 陳述式取得項目。
- 透過執行 INSERT 陳述式新增項目。
- 透過執行 UPDATE 陳述式更新項目。
- 透過執行 DELETE 陳述式刪除項目。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun main() {
    val ddb = DynamoDbClient.fromEnvironment { 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)
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ExecuteStatement。

使用適用於 Kotlin 的 SDK 的 Amazon EC2 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon EC2 來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 基本概念
- 動作

開始使用

您好 Amazon EC2

下列程式碼範例說明如何開始使用 Amazon EC2。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeSecurityGroups。

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立金鑰對和安全群組。
- 選取 Amazon Machine Image (AMI) 和相容的執行個體類型，然後建立執行個體。

- 停止並重新啟動執行個體。
- 將彈性 IP 地址與您的執行個體建立關聯。
- 使用 SSH 連線至執行個體，然後清理資源。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair

- CreateSecurityGroup
- DeleteKeyPair
- DeleteSecurityGroup
- DescribeImages
- DescribeInstanceTypes
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DisassociateAddress
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

動作

`AllocateAddress`

以下程式碼範例顯示如何使用 `AllocateAddress`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getAllocateAddress(instanceIdVal: String?): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }
```

```
    Ec2Client.fromEnvironment { 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
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 AllocateAddress。

`AssociateAddress`

以下程式碼範例顯示如何使用 `AssociateAddress`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }
```

```
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 AssociateAddress。

`AuthorizeSecurityGroupIngress`

以下程式碼範例顯示如何使用 `AuthorizeSecurityGroupIngress`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { 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
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 AuthorizeSecurityGroupIngress。

`CreateKeyPair`

以下程式碼範例顯示如何使用 `CreateKeyPair`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createEC2KeyPair(keyNameVal: String) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        println("The key ID is ${response.keyPairId}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateKeyPair。

`CreateSecurityGroup`

以下程式碼範例顯示如何使用 `CreateSecurityGroup`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createEC2SecurityGroup(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { 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
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateSecurityGroup。

DeleteKeyPair

以下程式碼範例顯示如何使用 DeleteKeyPair。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteKeys(keyPair: String?) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DeleteKeyPair。

`DeleteSecurityGroup`

以下程式碼範例顯示如何使用 `DeleteSecurityGroup`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteEC2SecGroup(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }
```

```
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted Security Group with id $groupIdVal")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DeleteSecurityGroup。

`DescribeInstanceTypes`

以下程式碼範例顯示如何使用 `DescribeInstanceTypes`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
// 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.fromEnvironment { 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
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DescribeInstanceTypes。

`DescribeInstances`

以下程式碼範例顯示如何使用 `DescribeInstances`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeEC2Instances() {
    val request =
        DescribeInstancesRequest {
            maxResults = 6
        }

    Ec2Client.fromEnvironment { 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}")
            }
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DescribeInstances。

`DescribeKeyPairs`

以下程式碼範例顯示如何使用 `DescribeKeyPairs`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeEC2Keys() {
    Ec2Client.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DescribeKeyPairs。

`DescribeSecurityGroups`

以下程式碼範例顯示如何使用 `DescribeSecurityGroups`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeSecurityGroups。

`DisassociateAddress`

以下程式碼範例顯示如何使用 `DisassociateAddress`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun disassociateAddressSc(associationIdVal: String?) {
```

```
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DisassociateAddress。

`ReleaseAddress`

以下程式碼範例顯示如何使用 `ReleaseAddress`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 ReleaseAddress。

`RunInstances`

以下程式碼範例顯示如何使用 `RunInstances`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createEC2Instance(
    name: String,
    amiId: String,
): String? {
    val request =
        RunInstancesRequest {
            imageId = amiId
            instanceType = InstanceType.T1Micro
            maxCount = 1
            minCount = 1
        }

    Ec2Client.fromEnvironment { 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
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 RunInstances。

`StartInstances`

以下程式碼範例顯示如何使用 `StartInstances`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { 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")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 StartInstances。

`StopInstances`

以下程式碼範例顯示如何使用 `StopInstances`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { 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")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 StopInstances。

`TerminateInstances`

以下程式碼範例顯示如何使用 `TerminateInstances`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun terminateEC2(instanceID: String) {
    val request =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceID)
        }

    Ec2Client.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 TerminateInstances。

使用適用於 Kotlin 的 SDK 的 Amazon ECR 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon ECR 來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 基本概念
- 動作

開始使用

Hello Amazon ECR

下列程式碼範例示範如何開始使用 Amazon ECR。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
        val imageResponse = ecrClient.listImages(listImages)
        imageResponse.imageIds?.forEach { imageId ->
            println("Image tag: ${imageId.imageTag}")
        }
    }
```

```
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 listImages。

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立 Amazon ECR 儲存庫。
- 設定儲存庫政策。
- 擷取儲存庫 URI。
- 取得 Amazon ECR 授權字符。
- 設定 Amazon ECR 儲存庫的生命週期政策。
- 將 Docker 映像檔推送至 Amazon ECR 儲存庫。
- 驗證 Amazon ECR 儲存庫中是否有映像存在。
- 列出您帳戶的 Amazon ECR 儲存庫，並取得其詳細資訊。
- 刪除 Amazon ECR 儲存庫。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

執行展示 Amazon ECR 功能的互動式情境。

```
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 方法的包裝函式類別。

```
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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)
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

動作

`CreateRepository`

以下程式碼範例顯示如何使用 `CreateRepository`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an empty
string if the operation failed.
     * @throws 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.fromEnvironment { 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
        }
    }
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateRepository。

DeleteRepository

以下程式碼範例顯示如何使用 DeleteRepository。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
    /**
     * 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.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteRepository。

`DescribeImages`

以下程式碼範例顯示如何使用 `DescribeImages`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    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.fromEnvironment { 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.")
```

```
            }
        }
    }
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeImages。

`DescribeRepositories`

以下程式碼範例顯示如何使用 `DescribeRepositories`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    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.fromEnvironment { 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 ""
            }
        }
    }
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DescribeRepositories。

`GetAuthorizationToken`

以下程式碼範例顯示如何使用 `GetAuthorizationToken`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient.fromEnvironment { 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.")
            }
        }
    }
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetAuthorizationToken。

`GetRepositoryPolicy`

以下程式碼範例顯示如何使用 `GetRepositoryPolicy`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
    /**
     * 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.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetRepositoryPolicy。

PushImageCmd

以下程式碼範例顯示如何使用 PushImageCmd。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    suspend fun pushDockerImage(
        repoName: String,
        imageName: String,
    ) {
        println("Pushing $imageName to $repoName will take a few seconds")
        val authConfig = getAuthConfig(repoName)

        EcrClient.fromEnvironment { 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)
            }
        }
    }
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 PushImageCmd。

`SetRepositoryPolicy`

以下程式碼範例顯示如何使用 `SetRepositoryPolicy`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 * Sets the repository policy for the specified ECR repository.
 *
 * @param repoName the name of the ECR repository.
 * @param iamRole the IAM role to be granted access to the repository.
 */
```

```
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.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
        val response = ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
        if (response != null) {
            println("Repository policy set successfully.")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 SetRepositoryPolicy。

StartLifecyclePolicyPreview

以下程式碼範例顯示如何使用 StartLifecyclePolicyPreview。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    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.fromEnvironment { 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.")
```

```
            }
        }
    }
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 StartLifecyclePolicyPreview。

使用適用於 Kotlin 的 SDK 的 OpenSearch Service 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 OpenSearch Service 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作

動作

`CreateDomain`

以下程式碼範例顯示如何使用 `CreateDomain`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        val createResponse = searchClient.createDomain(request)
        println("Domain status is ${createResponse.domainStatus}")
        println("Domain Id is ${createResponse.domainStatus?.domainId}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateDomain。

DeleteDomain

以下程式碼範例顯示如何使用 DeleteDomain。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteSpecificDomain(domainNameVal: String) {
    val request =
        DeleteDomainRequest {
            domainName = domainNameVal
        }
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        searchClient.deleteDomain(request)
        println("$domainNameVal was successfully deleted.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteDomain。

`ListDomainNames`

以下程式碼範例顯示如何使用 `ListDomainNames`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listAllDomains() {
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        val response: ListDomainNamesResponse =
 searchClient.listDomainNames(ListDomainNamesRequest {})
        response.domainNames?.forEach { domain ->
```

```
            println("Domain name is " + domain.domainName)
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListDomainNames。

`UpdateDomainConfig`

以下程式碼範例顯示如何使用 `UpdateDomainConfig`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun updateSpecificDomain(domainNameVal: String?) {
    val clusterConfigOb =
        ClusterConfig {
            instanceCount = 3
        }

    val request =
        UpdateDomainConfigRequest {
            domainName = domainNameVal
            clusterConfig = clusterConfigOb
        }

    println("Sending domain update request...")
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        val updateResponse = searchClient.updateDomainConfig(request)
        println("Domain update response from Amazon OpenSearch Service:")
        println(updateResponse.toString())
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 UpdateDomainConfig。

使用適用於 Kotlin 的 SDK 的 EventBridge 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 EventBridge 來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 基本概念
- 動作

開始使用

您好 EventBridge

下列程式碼範例說明如何開始使用 EventBridge。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListEventBuses。

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立規則並在其中新增目標。
- 啟用和停用規則。
- 列出並更新規則和目標。
- 發送事件，然後清理資源。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/*
 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println(" $topicArnVal was deleted.")
    }
}

suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
    EventBridgeClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { s3Client ->
        s3Client.putObject(putOb)
    }
}

suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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"
            }
        ]
    }
    """.trimIndent()

    val topicAttributes = mutableMapOf<String, String>()
    topicAttributes["Policy"] = topicPolicy

    val topicRequest =
        CreateTopicRequest {
            name = topicName
            attributes = topicAttributes
        }
```

```
    SnsClient.fromEnvironment { 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.fromEnvironment { 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"]
            }
        }
    }
    """.trimIndent()
```

```
    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }

    EventBridgeClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { iam ->
        val response = iam.createRole(request)
        iam.attachRolePolicy(rolePolicyRequest)
        return response.role?.arn
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

動作

`DeleteRule`

以下程式碼範例顯示如何使用 `DeleteRule`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteRuleByName(ruleName: String?) {
```

```
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteRule。

`DescribeRule`

以下程式碼範例顯示如何使用 `DescribeRule`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeRule。

`DisableRule`

以下程式碼範例顯示如何使用 `DisableRule`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DisableRule。

`EnableRule`

以下程式碼範例顯示如何使用 `EnableRule`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 EnableRule。

`ListRuleNamesByTarget`

以下程式碼範例顯示如何使用 `ListRuleNamesByTarget`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListRuleNamesByTarget。

`ListRules`

以下程式碼範例顯示如何使用 `ListRules`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listRules() {
    val rulesRequest =
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListRules。

`ListTargetsByRule`

以下程式碼範例顯示如何使用 `ListTargetsByRule`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListTargetsByRule。

`PutEvents`

以下程式碼範例顯示如何使用 `PutEvents`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putEvents(eventsRequest)
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 PutEvents。

`PutRule`

以下程式碼範例顯示如何使用 `PutRule`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

建立排程規則。

```
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.fromEnvironment { region = "us-west-2" }.use { eventBrClient -
>
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

建立在物件新增至 Amazon Simple Storage Service 儲存貯體時觸發的規則。

```
// 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"]
            }
        }
    }
    """.trimIndent()

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
```

```
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 PutRule。

`PutTargets`

以下程式碼範例顯示如何使用 `PutTargets`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
// 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.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target $topicName
 for bucket $bucketName.")
    }
}
```

將輸入轉換器新增至某個規則的目標。

```
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.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(targetsRequest)
    }
```

```
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 PutTargets。

`RemoveTargets`

以下程式碼範例顯示如何使用 `RemoveTargets`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient.fromEnvironment { 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")
            }
        }
```

```
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 RemoveTargets。

AWS Glue使用適用於 Kotlin 的 SDK 的範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 來執行動作和實作常見案例AWS Glue。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 基本概念
- 動作

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立網路爬取公有 Amazon S3 儲存貯體的爬蟲程式，以及產生 CSV 格式中繼資料的資料庫。
- 列出 中資料庫和資料表的相關資訊AWS Glue Data Catalog。
- 建立從 S3 儲存貯體中擷取 CSV 資料的任務、轉換資料，以及將 JSON 格式的輸出載入至另一個 S3 儲存貯體。
- 列出任務執行的相關資訊、檢視已轉換的資料以及清除資源。

如需詳細資訊，請參閱教學課程：AWS Glue Studio 入門。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.createJob(jobRequest)
        println("$jobName was successfully created.")
    }
}

suspend fun getJobs() {
    val request =
        GetJobsRequest {
```

```
            maxResults = 10
        }

    GlueClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

動作

`CreateCrawler`

以下程式碼範例顯示如何使用 `CreateCrawler`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-west-2" }.use { glueClient ->
        glueClient.createCrawler(request)
        println("$crawlerName was successfully created")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的 CreateCrawler。

`GetCrawler`

以下程式碼範例顯示如何使用 `GetCrawler`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getSpecificCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }
    GlueClient.fromEnvironment { 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")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的 GetCrawler。

`GetDatabase`

以下程式碼範例顯示如何使用 `GetDatabase`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getSpecificDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的 GetDatabase。

`StartCrawler`

以下程式碼範例顯示如何使用 `StartCrawler`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun startSpecificCrawler(crawlerName: String?) {
    val request =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient.fromEnvironment { region = "us-west-2" }.use { glueClient ->
        glueClient.startCrawler(request)
        println("$crawlerName was successfully started.")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的 StartCrawler。

使用適用於 Kotlin 的 SDK 的 IAM 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 IAM 來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 基本概念
- 動作

基本概念

了解基本概念

下列程式碼範例示範如何建立使用者並擔任角色。

> **Warning**
>
> 為避免安全風險，在開發專用軟體或使用真實資料時，請勿使用 IAM 使用者進行身分驗證。相反地，搭配使用聯合功能和身分提供者，例如 AWS IAM Identity Center。

- 建立沒有許可的使用者。
- 建立一個可授予許可的角色，以列出帳戶的 Amazon S3 儲存貯體。
- 新增政策，讓使用者擔任該角色。
- 使用暫時憑證，擔任角色並列出 S3 儲存貯體，然後清理資源。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

建立可包裝 IAM 使用者動作的函數。

```
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 = """
```

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

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment {
        region = "us-east-1"
        credentialsProvider = staticCredentials
    }

    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.fromEnvironment { 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.fromEnvironment { 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)
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的下列主題。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

動作

`AttachRolePolicy`

以下程式碼範例顯示如何使用 `AttachRolePolicy`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun attachIAMRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient.fromEnvironment { 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
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的 AttachRolePolicy。

`CreateAccessKey`

以下程式碼範例顯示如何使用 `CreateAccessKey`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createIAMAccessKey(user: String?): String {
    val request =
        CreateAccessKeyRequest {
            userName = user
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的 CreateAccessKey。

`CreateAccountAlias`

以下程式碼範例顯示如何使用 `CreateAccountAlias`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createIAMAccountAlias(alias: String) {
    val request =
        CreateAccountAliasRequest {
            accountAlias = alias
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的 CreateAccountAlias。

`CreatePolicy`

以下程式碼範例顯示如何使用 `CreatePolicy`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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": "*"
            }
        ]
    }
    """.trimIndent()

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的 CreatePolicy。

`CreateUser`

以下程式碼範例顯示如何使用 `CreateUser`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createIAMUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的 CreateUser。

`DeleteAccessKey`

以下程式碼範例顯示如何使用 `DeleteAccessKey`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteKey(
    userNameVal: String,
    accessKey: String,
) {
```

```
    val request =
        DeleteAccessKeyRequest {
            accessKeyId = accessKey
            userName = userNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccessKey(request)
        println("Successfully deleted access key $accessKey from $userNameVal")
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的 DeleteAccessKey。

`DeleteAccountAlias`

以下程式碼範例顯示如何使用 `DeleteAccountAlias`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteIAMAccountAlias(alias: String) {
    val request =
        DeleteAccountAliasRequest {
            accountAlias = alias
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的 DeleteAccountAlias。

`DeletePolicy`

以下程式碼範例顯示如何使用 `DeletePolicy`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {
    val request =
        DeletePolicyRequest {
            policyArn = policyARNVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的 DeletePolicy。

`DeleteUser`

以下程式碼範例顯示如何使用 `DeleteUser`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的 DeleteUser。

`DetachRolePolicy`

以下程式碼範例顯示如何使用 `DetachRolePolicy`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun detachPolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        DetachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")
    }
```

```
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的 DetachRolePolicy。

`GetPolicy`

以下程式碼範例顯示如何使用 `GetPolicy`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getIAMPolicy(policyArnVal: String?) {
    val request =
        GetPolicyRequest {
            policyArn = policyArnVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的 GetPolicy。

`ListAccessKeys`

以下程式碼範例顯示如何使用 `ListAccessKeys`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listKeys(userNameVal: String?) {
    val request =
        ListAccessKeysRequest {
            userName = userNameVal
        }
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的 ListAccessKeys。

`ListAccountAliases`

以下程式碼範例顯示如何使用 `ListAccountAliases`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listAliases() {
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
```

```
            println("Retrieved account alias $alias")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的 ListAccountAliases。

`ListUsers`

以下程式碼範例顯示如何使用 `ListUsers`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listAllUsers() {
    IamClient.fromEnvironment { 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}")
            }
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的 ListUsers。

`UpdateUser`

以下程式碼範例顯示如何使用 `UpdateUser`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun updateIAMUser(
    curName: String?,
    newName: String?,
) {
    val request =
        UpdateUserRequest {
            userName = curName
            newUserName = newName
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.updateUser(request)
        println("Successfully updated user to $newName")
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的 UpdateUser。

AWS IoT使用適用於 Kotlin 的 SDK 的範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 來執行動作和實作常見案例AWS IoT。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用

- 基本概念
- 動作

開始使用

您好AWS IoT

下列程式碼範例示範如何開始使用AWS IoT。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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}")
            }
        }
    }
```

```
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 listThings。

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立AWS IoT物件。
- 產生裝置憑證。
- 使用 屬性更新AWS IoT物件。
- 傳回唯一的端點。
- 列出您的AWS IoT憑證。
- 更新影AWS IoT子。
- 寫出狀態資訊。
- 建立規則。
- 列出您的規則。
- 使用 Thing 名稱搜尋物件。
- 刪除AWS IoT物件。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
```

```
        println("Deleted $thingNameVal")
    }
}

suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
```

```
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings
  - SearchIndex
  - UpdateIndexingConfiguration
  - UpdateThing

動作

`AttachThingPrincipal`

以下程式碼範例顯示如何使用 `AttachThingPrincipal`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 AttachThingPrincipal。

`CreateKeysAndCertificate`

以下程式碼範例顯示如何使用 `CreateKeysAndCertificate`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { 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
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateKeysAndCertificate。

`CreateThing`

以下程式碼範例顯示如何使用 `CreateThing`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateThing。

`CreateTopicRule`

以下程式碼範例顯示如何使用 `CreateTopicRule`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateTopicRule。

`DeleteCertificate`

以下程式碼範例顯示如何使用 `DeleteCertificate`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteCertificate。

`DeleteThing`

以下程式碼範例顯示如何使用 `DeleteThing`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteThing。

`DescribeEndpoint`

以下程式碼範例顯示如何使用 `DescribeEndpoint`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { 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
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DescribeEndpoint。

`DescribeThing`

以下程式碼範例顯示如何使用 `DescribeThing`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeThing。

`DetachThingPrincipal`

以下程式碼範例顯示如何使用 `DetachThingPrincipal`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS適用於 Kotlin 的 SDK API 參考》中的 DetachThingPrincipal。

`ListCertificates`

以下程式碼範例顯示如何使用 `ListCertificates`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listCertificates() {
    IotClient.fromEnvironment { 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}")
        }
    }
```

```
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListCertificates。

`SearchIndex`

以下程式碼範例顯示如何使用 `SearchIndex`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { 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}") }
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 SearchIndex。

`UpdateThing`

以下程式碼範例顯示如何使用 `UpdateThing`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 UpdateThing。

AWS IoT data使用適用於 Kotlin 的 SDK 的範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 來執行動作和實作常見案例AWS IoT data。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作

動作

`GetThingShadow`

以下程式碼範例顯示如何使用 `GetThingShadow`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient.fromEnvironment { 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")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetThingShadow。

`UpdateThingShadow`

以下程式碼範例顯示如何使用 UpdateThingShadow。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { iotPlaneClient
 ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 UpdateThingShadow。

使用適用於 Kotlin 的 SDK 的 Amazon Keyspaces 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon Keyspaces 來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 基本概念
- 動作

開始使用

Hello Amazon Keyspaces

下列程式碼範例說明如何開始使用 Amazon Keyspaces。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.listKeyspaces(keyspacesRequest)
        response.keyspaces?.forEach { keyspace ->
            println("The name of the keyspace is ${keyspace.keyspaceName}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListKeyspaces。

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立鍵空間和資料表。資料表結構描述會保留電影資料，並啟用時間點復原。
- 使用具有 SigV4 驗證的安全 TLS 連線，以連接至鍵空間。
- 查詢資料表。新增、擷取和更新電影資料。
- 更新資料表。新增資料欄以追蹤觀看的電影。
- 將資料表還原至其先前的狀態，並清除資源。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
```

```
 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}

suspend fun listKeyspacesPaginator() {
    KeyspacesClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

動作

`CreateKeyspace`

以下程式碼範例顯示如何使用 `CreateKeyspace`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
```

```
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateKeyspace。

`CreateTable`

以下程式碼範例顯示如何使用 `CreateTable`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateTable。

`DeleteKeyspace`

以下程式碼範例顯示如何使用 `DeleteKeyspace`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteKeyspace。

`DeleteTable`

以下程式碼範例顯示如何使用 `DeleteTable`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteTable。

`GetKeyspace`

以下程式碼範例顯示如何使用 `GetKeyspace`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
```

```
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse = keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetKeyspace。

`GetTable`

以下程式碼範例顯示如何使用 `GetTable`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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}")
            }
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetTable。

`ListKeyspaces`

以下程式碼範例顯示如何使用 `ListKeyspaces`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listKeyspacesPaginator() {
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
```

```
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListKeyspaces。

`ListTables`

以下程式碼範例顯示如何使用 `ListTables`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { 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}")
            }
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListTables。

`RestoreTable`

以下程式碼範例顯示如何使用 `RestoreTable`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 RestoreTable。

`UpdateTable`

以下程式碼範例顯示如何使用 `UpdateTable`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 UpdateTable。

AWS KMS使用適用於 Kotlin 的 SDK 的範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 來執行動作和實作常見案例AWS KMS。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作

動作

`CreateAlias`

以下程式碼範例顯示如何使用 `CreateAlias`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createCustomAlias(
    targetKeyIdVal: String?,
    aliasNameVal: String?,
) {
    val request =
        CreateAliasRequest {
            aliasName = aliasNameVal
            targetKeyId = targetKeyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.createAlias(request)
        println("$aliasNameVal was successfully created")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateAlias。

`CreateGrant`

以下程式碼範例顯示如何使用 `CreateGrant`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.createGrant(request)
        return response.grantId
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateGrant。

`CreateKey`

以下程式碼範例顯示如何使用 `CreateKey`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createKey(keyDesc: String?): String? {
    val request =
        CreateKeyRequest {
            description = keyDesc
            customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
            keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
        }

    KmsClient.fromEnvironment { 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
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateKey。

`Decrypt`

以下程式碼範例顯示如何使用 `Decrypt`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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)
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 Decrypt。

`DescribeKey`

以下程式碼範例顯示如何使用 `DescribeKey`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeSpecifcKey(keyIdVal: String?) {
    val request =
        DescribeKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { 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}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeKey。

`DisableKey`

以下程式碼範例顯示如何使用 `DisableKey`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun disableKey(keyIdVal: String?) {
    val request =
        DisableKeyRequest {
            keyId = keyIdVal
```

```
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.disableKey(request)
        println("$keyIdVal was successfully disabled")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DisableKey。

`EnableKey`

以下程式碼範例顯示如何使用 `EnableKey`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun enableKey(keyIdVal: String?) {
    val request =
        EnableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.enableKey(request)
        println("$keyIdVal was successfully enabled.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 EnableKey。

`Encrypt`

以下程式碼範例顯示如何使用 `Encrypt`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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)
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 Encrypt。

`ListAliases`

以下程式碼範例顯示如何使用 `ListAliases`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listAllAliases() {
    val request =
        ListAliasesRequest {
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listAliases(request)
        response.aliases?.forEach { alias ->
            println("The alias name is ${alias.aliasName}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListAliases。

`ListGrants`

以下程式碼範例顯示如何使用 `ListGrants`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun displayGrantIds(keyIdVal: String?) {
    val request =
        ListGrantsRequest {
            keyId = keyIdVal
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listGrants(request)
        response.grants?.forEach { grant ->
            println("The grant Id is ${grant.grantId}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListGrants。

`ListKeys`

以下程式碼範例顯示如何使用 `ListKeys`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listAllKeys() {
    val request =
```

```
        ListKeysRequest {
            limit = 15
        }

    KmsClient.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListKeys。

使用適用於 Kotlin 的 SDK 的 Lambda 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Lambda 來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 基本概念
- 動作
- 案例

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立 IAM 角色和 Lambda 函數，然後上傳處理常式程式碼。

- 調用具有單一參數的函數並取得結果。
- 更新函數程式碼並使用環境變數進行設定。
- 調用具有新參數的函數並取得結果。顯示傳回的執行日誌。
- 列出您帳戶的函數，然後清理相關資源。

如需詳細資訊，請參閱使用主控台建立 Lambda 函數。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode

- UpdateFunctionConfiguration

動作

`CreateFunction`

以下程式碼範例顯示如何使用 `CreateFunction`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS開發套件 API 參考》中的「CreateFunction」。

`DeleteFunction`

以下程式碼範例顯示如何使用 `DeleteFunction`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun delLambdaFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS開發套件 API 參考》中的「DeleteFunction」。

Invoke

以下程式碼範例顯示如何使用 Invoke。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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}")
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS開發套件 API 參考》中的「Invoke」。

案例

建立無伺服器應用程式來管理相片

下列程式碼範例示範如何建立無伺服器應用程式，讓使用者以標籤管理相片。

SDK for Kotlin

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

使用適用於 Kotlin 的 SDK 的 Amazon Location 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon Location 來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 基本概念
- 動作

開始使用

Hello Amazon Location

下列程式碼範例示範如何開始使用 Amazon Location Service。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
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.fromEnvironment { 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}")
            }
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - ListGeofenceCollections
  - ListGeofences

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立 Amazon Location 地圖。
- 建立 Amazon Location API 金鑰。
- 顯示地圖 URL。

- 建立地理柵欄集合。
- 儲存地理柵欄幾何。
- 建立追蹤器資源。
- 更新裝置的定位。
- 擷取指定裝置的最新定位更新。
- 建立路線計算器。
- 判斷西雅圖和溫哥華之間的距離。
- 使用 Amazon Location 更高階 API。
- 刪除 Amazon Location 資產。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - BatchUpdateDevicePosition
  - CalculateRoute
  - CreateGeofenceCollection
  - CreateKey
  - CreateMap
  - CreateRouteCalculator
  - CreateTracker
  - DeleteGeofenceCollection
  - DeleteKey
  - DeleteMap
  - DeleteRouteCalculator
  - DeleteTracker
  - GetDevicePosition
  - PutGeofence

動作

`BatchUpdateDevicePosition`

以下程式碼範例顯示如何使用 `BatchUpdateDevicePosition`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 * Updates the position of a device in the location tracking system.
 *
 * @param trackerName the name of the tracker associated with the device
 * @param deviceId    the unique identifier of the device
 */
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.fromEnvironment { region = "us-east-1" }.use { client ->
        client.batchUpdateDevicePosition(request)
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 BatchUpdateDevicePosition。

`CalculateRoute`

以下程式碼範例顯示如何使用 `CalculateRoute`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 * 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.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.calculateRoute(request)
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CalculateRoute。

`CreateGeofenceCollection`

以下程式碼範例顯示如何使用 `CreateGeofenceCollection`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 * 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.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createGeofenceCollection(collectionRequest)
        return response.collectionArn
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateGeofenceCollection。

`CreateKey`

以下程式碼範例顯示如何使用 `CreateKey`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 * Creates a new API key with the specified name and restrictions.
 *
 * @param keyName the name of the API key to be created
 * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which the
 API key will be associated
 * @return 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.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createKey(request)
        return response.keyArn
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateKey。

`CreateMap`

以下程式碼範例顯示如何使用 `CreateMap`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 * 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.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createMap(mapRequest)
        return response.mapArn
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateMap。

`CreateRouteCalculator`

以下程式碼範例顯示如何使用 `CreateRouteCalculator`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 * 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.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.createRouteCalculator(request)
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateRouteCalculator。

`CreateTracker`

以下程式碼範例顯示如何使用 `CreateTracker`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 * Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
 *
 * @param trackerName the name of the tracker to be created
 * @return a {@link CompletableFuture} that, when completed, will contain the Amazon
 Resource Name (ARN) of the created tracker
 */
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.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createTracker(trackerRequest)
        return response.trackerArn
```

```
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateTracker。

`DeleteGeofenceCollection`

以下程式碼範例顯示如何使用 `DeleteGeofenceCollection`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 * 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.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteGeofenceCollection(collectionRequest)
        println("The geofence collection $collectionName was deleted.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteGeofenceCollection。

DeleteKey

以下程式碼範例顯示如何使用 DeleteKey。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 * 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.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteKey(keyRequest)
        println("The key $keyName was deleted.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteKey。

DeleteMap

以下程式碼範例顯示如何使用 DeleteMap。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 * 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.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteMap(mapRequest)
        println("The map $mapName was deleted.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteMap。

`DeleteRouteCalculator`

以下程式碼範例顯示如何使用 `DeleteRouteCalculator`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
```

```
 * 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.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteRouteCalculator(calculatorRequest)
        println("The route calculator $calcName was deleted.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteRouteCalculator。

`DeleteTracker`

以下程式碼範例顯示如何使用 `DeleteTracker`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 * 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.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》的 DeleteTracker。

`GetDevicePosition`

以下程式碼範例顯示如何使用 `GetDevicePosition`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 * Retrieves the position of a device using the provided LocationClient.
 *
 * @param trackerName The name of the tracker associated with the device.
 * @param deviceId    The ID of the device to retrieve the position for.
 */
suspend fun getDevicePosition(trackerName: String, deviceId: String):
 GetDevicePositionResponse {
    val request = GetDevicePositionRequest {
        this.trackerName = trackerName
        this.deviceId = deviceId
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.getDevicePosition(request)
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetDevicePosition。

PutGeofence

以下程式碼範例顯示如何使用 PutGeofence。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 * Adds a new geofence to the specified collection.
 *
 * @param collectionName the name of the geofence collection to add the geofence to
 * @param geoId          the unique identifier for the geofence
 */
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.fromEnvironment { region = "us-east-1" }.use { client ->
        client.putGeofence(geofenceRequest)
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 PutGeofence。

使用適用於 Kotlin 的 SDK 的 MediaConvert 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 MediaConvert 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作

動作

`CreateJob`

以下程式碼範例顯示如何使用 `CreateJob`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createMediaJob(
    mcClient: MediaConvertClient,
    mcRoleARN: String,
    fileInput1: String,
): String? {
    // Step 1: Describe endpoints to get the MediaConvert endpoint URL
    val describeResponse = mcClient.describeEndpoints(
        DescribeEndpointsRequest {
            maxResults = 1
        },
    )
```

```
val endpointUrl = describeResponse.endpoints?.firstOrNull()?.url
    ?: error("No MediaConvert endpoint found")

// Step 2: Create MediaConvert client with resolved endpoint
val mediaConvert = MediaConvertClient.fromEnvironment {
    region = "us-west-2"
    endpointProvider = MediaConvertEndpointProvider {
        Endpoint(endpointUrl)
    }
}

// Output destination folder in S3 - put in 'output/' folder beside input
val outputDestination = fileInput1.substringBeforeLast('/') + "/output/"

// Step 3: Create the job request with minimal valid video codec settings
val jobRequest = CreateJobRequest {
    role = mcRoleARN
    settings = JobSettings {
        inputs = listOf(
            Input {
                fileInput = fileInput1
            },
        )
        outputGroups = listOf(
            OutputGroup {
                outputGroupSettings = OutputGroupSettings {
                    type = OutputGroupType.FileGroupSettings
                    fileGroupSettings = FileGroupSettings {
                        destination = outputDestination
                    }
                }
                outputs = listOf(
                    Output {
                        containerSettings = ContainerSettings {
                            container = ContainerType.Mp4
                        }
                        videoDescription = VideoDescription {
                            width = 1280
                            height = 720
                            codecSettings = VideoCodecSettings {
                                codec = VideoCodec.H264
                                h264Settings = H264Settings {
                                    rateControlMode = H264RateControlMode.Qvbr
```

```
                                    qvbrSettings = H264QvbrSettings {
                                        qvbrQualityLevel = 7
                                    }
                                    maxBitrate = 5_000_000
                                    codecLevel = H264CodecLevel.Auto
                                    codecProfile = H264CodecProfile.Main
                                    framerateControl =
 H264FramerateControl.InitializeFromSource
                                }
                            }
                        }
                    },
                )
            },
        )
    }
}

    // Step 4: Call MediaConvert to create the job
    val response = mediaConvert.createJob(jobRequest)

    // Return the job ID or null if not found
    return response.job?.id
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateJob。

`GetJob`

以下程式碼範例顯示如何使用 `GetJob`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getSpecificJob(mcClient: MediaConvertClient, jobId: String) {
```

```
    // 1. Discover the correct endpoint
    val res = mcClient.describeEndpoints(DescribeEndpointsRequest { maxResults =
 1 })
    var endpointUrl = res.endpoints?.firstOrNull()?.url
        ?: error(" No MediaConvert endpoint found")

    // 2. Create a new client using the endpoint
    val clientWithEndpoint = MediaConvertClient {
        region = "us-west-2"
        endpointUrl = endpointUrl
    }

    // 3. Get the job details
    val jobResponse = clientWithEndpoint.getJob(GetJobRequest { id = jobId })
    val job = jobResponse.job

    println("Job status: ${job?.status}")
    println("Job ARN: ${job?.arn}")
    println("Output group count: ${job?.settings?.outputGroups?.size}")
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetJob。

`ListJobs`

以下程式碼範例顯示如何使用 `ListJobs`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListJobs。

使用適用於 Kotlin 的 SDK 的 Amazon Pinpoint 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon Pinpoint 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作

動作

CreateApp

以下程式碼範例顯示如何使用 CreateApp。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createApplication(applicationName: String?): String? {
    val createApplicationRequestOb =
        CreateApplicationRequest {
            name = applicationName
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.createApp(
                CreateAppRequest {
                    createApplicationRequest = createApplicationRequestOb
                },
            )
        return result.applicationResponse?.id
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API 參考中的 CreateApp。

CreateCampaign

以下程式碼範例顯示如何使用 CreateCampaign。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result: CreateCampaignResponse =
            pinpoint.createCampaign(
                CreateCampaignRequest {
```

```
                    applicationId = appId
                    writeCampaignRequest = writeCampaign
                },
            )
        println("Campaign ID is ${result.campaignResponse?.id}")
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API 參考中的 CreateCampaign。

`CreateSegment`

以下程式碼範例顯示如何使用 `CreateSegment`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API 參考中的 CreateSegment。

DeleteApp

以下程式碼範例顯示如何使用 DeleteApp。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deletePinApp(appId: String?) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.deleteApp(
                DeleteAppRequest {
                    applicationId = appId
                },
            )
        val appName = result.applicationResponse?.name
        println("Application $appName has been deleted.")
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API 參考中的 DeleteApp。

`DeleteEndpoint`

以下程式碼範例顯示如何使用 `DeleteEndpoint`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deletePinEncpoint(
    appIdVal: String?,
    endpointIdVal: String?,
) {
```

```
    val deleteEndpointRequest =
        DeleteEndpointRequest {
            applicationId = appIdVal
            endpointId = endpointIdVal
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.deleteEndpoint(deleteEndpointRequest)
        val id = result.endpointResponse?.id
        println("The deleted endpoint is  $id")
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API 參考中的 DeleteEndpoint。

`GetEndpoint`

以下程式碼範例顯示如何使用 `GetEndpoint`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun lookupPinpointEndpoint(
    appId: String?,
    endpoint: String?,
) {
    PinpointClient.fromEnvironment { 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)
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API 參考中的 GetEndpoint。

`GetSegments`

以下程式碼範例顯示如何使用 `GetSegments`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listSegs(appId: String?) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val response =
            pinpoint.getSegments(
                GetSegmentsRequest {
                    applicationId = appId
                },
            )
        response.segmentsResponse?.item?.forEach { segment ->
            println("Segement id is ${segment.id}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API 參考中的 GetSegments。

SendMessages

以下程式碼範例顯示如何使用 SendMessages。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
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.fromEnvironment { region = "us-east-1" }.use { pinpointemail
 ->
        pinpointemail.sendEmail(sendEmailRequest)
        println("Message Sent")
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API 參考中的 SendMessages。

使用適用於 Kotlin 的 SDK 的 Amazon RDS 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon RDS 來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 基本概念
- 動作

- 案例

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立自訂資料庫參數群組並設定參數值。
- 建立資料庫執行個體，設定為使用參數群組。資料庫執行個體也包含資料庫。
- 擷取執行個體的快照。
- 刪除執行個體和參數群組。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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 = "8.4"
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-west-2" }.use
 { secretsClient ->
        val valueResponse = secretsClient.getSecretValue(valueRequest)
        return valueResponse.secretString
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

動作

`CreateDBInstance`

以下程式碼範例顯示如何使用 `CreateDBInstance`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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"
            engineVersion = "8.4"
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { 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.fromEnvironment { 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!")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateDBInstance。

`DeleteDBInstance`

以下程式碼範例顯示如何使用 `DeleteDBInstance`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteDatabaseInstance(dbInstanceIdentifierVal: String?) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteDBInstance。

`DescribeAccountAttributes`

以下程式碼範例顯示如何使用 `DescribeAccountAttributes`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getAccountAttributes() {
    RdsClient.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeAccountAttributes。

`DescribeDBInstances`

以下程式碼範例顯示如何使用 `DescribeDBInstances`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeInstances() {
    RdsClient.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeDBInstances。

`ModifyDBInstance`

以下程式碼範例顯示如何使用 `ModifyDBInstance`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun updateIntance(
    dbInstanceIdentifierVal: String?,
    masterUserPasswordVal: String?,
) {
    val request =
        ModifyDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            publiclyAccessible = true
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val instanceResponse = rdsClient.modifyDbInstance(request)
        println("The ARN of the modified database is
 ${instanceResponse.dbInstance?.dbInstanceArn}")
    }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ModifyDBInstance。

案例

建立 Aurora 無伺服器工作項目追蹤器

下列程式碼示範如何建立一個 Web 應用程式，該應用程式會追蹤 Amazon Aurora Serverless 資料庫中的工作項目，並使用 Amazon Simple Email Service (Amazon SES) 傳送報告。

SDK for Kotlin

說明如何建立可追蹤和報告存放在 Amazon RDS 資料庫中的工作項目的 Web 應用程式。

如需完整的原始碼以及如何設定 Spring REST API 以查詢 Amazon Aurora 無伺服器資料並供 React 應用程式使用的說明，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Aurora
- Amazon RDS
- Amazon RDS 資料服務
- Amazon SES

使用適用於 Kotlin 的 SDK 的 Amazon RDS Data Service 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon RDS Data Service 來執行動作和實作常見案例。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 案例

案例

建立 Aurora 無伺服器工作項目追蹤器

下列程式碼示範如何建立一個 Web 應用程式，該應用程式會追蹤 Amazon Aurora Serverless 資料庫中的工作項目，並使用 Amazon Simple Email Service (Amazon SES) 傳送報告。

SDK for Kotlin

說明如何建立可追蹤和報告存放在 Amazon RDS 資料庫中的工作項目的 Web 應用程式。

如需完整的原始碼以及如何設定 Spring REST API 以查詢 Amazon Aurora 無伺服器資料並供 React 應用程式使用的說明，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Aurora
- Amazon RDS
- Amazon RDS 資料服務
- Amazon SES

使用適用於 Kotlin 的 SDK 的 Amazon Redshift 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon Redshift 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作
- 案例

動作

`CreateCluster`

以下程式碼範例顯示如何使用 `CreateCluster`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

建立 叢集

```
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.fromEnvironment { region = "us-east-1" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.createCluster(clusterRequest)
        println("Created cluster ${clusterResponse.cluster?.clusterIdentifier}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateCluster。

`DeleteCluster`

以下程式碼範例顯示如何使用 `DeleteCluster`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

刪除叢集。

```
suspend fun deleteRedshiftCluster(clusterId: String?) {
    val request =
```

```
        DeleteClusterRequest {
            clusterIdentifier = clusterId
            skipFinalClusterSnapshot = true
        }

    RedshiftClient.fromEnvironment { region = "us-west-2" }.use { redshiftClient ->
        val response = redshiftClient.deleteCluster(request)
        println("The status is ${response.cluster?.clusterStatus}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteCluster。

`DescribeClusters`

以下程式碼範例顯示如何使用 `DescribeClusters`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

描述叢集。

```
suspend fun describeRedshiftClusters() {
    RedshiftClient.fromEnvironment { 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}")
            }
        }
    }
```

```
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DescribeClusters。

`ModifyCluster`

以下程式碼範例顯示如何使用 `ModifyCluster`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

修改叢集。

```
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",
        )
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ModifyCluster。

案例

建立用於追蹤 Amazon Redshift 資料的 Web 應用程式

下列程式碼範例示範如何使用 Amazon Redshift 資料庫，建立可追蹤和報告工作項目的 Web 應用程式。

SDK for Kotlin

說明如何建立可追蹤和報告存放在 Amazon Redshift 資料庫中的工作項目的 Web 應用程式。

如需完整的原始碼以及如何設定 Spring REST API 以查詢 Amazon Redshift 資料並供 React 應用程式使用的說明，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Amazon Redshift
- Amazon SES

使用適用於 Kotlin 的 SDK 的 Amazon Rekognition 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon Rekognition 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作
- 案例

動作

`CompareFaces`

以下程式碼範例顯示如何使用 `CompareFaces`。

如需詳細資訊，請參閱比較映像中的人臉。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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}")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS適用於 Kotlin 的 SDK API 參考》中的 CompareFaces。

`CreateCollection`

以下程式碼範例顯示如何使用 `CreateCollection`。

如需更多資訊，請參閱建立集合。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createMyCollection(collectionIdVal: String) {
    val request =
        CreateCollectionRequest {
```

```
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS適用於 Kotlin 的 SDK API 參考》中的 CreateCollection。

`DeleteCollection`

以下程式碼範例顯示如何使用 `DeleteCollection`。

如需更多資訊，請參閱刪除集合。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteMyCollection(collectionIdVal: String) {
    val request =
        DeleteCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS適用於 Kotlin 的 SDK API 參考》中的 DeleteCollection。

DeleteFaces

以下程式碼範例顯示如何使用 DeleteFaces。

如需詳細資訊，請參閱從集合中刪除人臉。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteFacesCollection(
    collectionIdVal: String?,
    faceIdVal: String,
) {
    val deleteFacesRequest =
        DeleteFacesRequest {
            collectionId = collectionIdVal
            faceIds = listOf(faceIdVal)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
        println("$faceIdVal was deleted from the collection")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS適用於 Kotlin 的 SDK API 參考》中的 DeleteFaces。

DescribeCollection

以下程式碼範例顯示如何使用 DescribeCollection。

如需詳細資訊，請參閱描述集合。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeColl(collectionName: String) {
    val request =
        DescribeCollectionRequest {
            collectionId = collectionName
        }

    RekognitionClient.fromEnvironment { 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}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Kotlin 的 SDK API 參考》中的 DescribeCollection。

`DetectFaces`

以下程式碼範例顯示如何使用 `DetectFaces`。

如需詳細資訊，請參閱偵測影像中的人臉。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun detectFacesinImage(sourceImage: String?) {
    val souImage =
```

```
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectFacesRequest {
            attributes = listOf(Attribute.All)
            image = souImage
        }

    RekognitionClient.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS適用於 Kotlin 的 SDK API 參考》中的 DetectFaces。

`DetectLabels`

以下程式碼範例顯示如何使用 `DetectLabels`。

如需詳細資訊，請參閱偵測映像中的標籤。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun detectImageLabels(sourceImage: String) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
```

```
        }
    val request =
        DetectLabelsRequest {
            image = souImage
            maxLabels = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectLabels(request)
        response.labels?.forEach { label ->
            println("${label.name} : ${label.confidence}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS適用於 Kotlin 的 SDK API 參考》中的 DetectLabels。

`DetectModerationLabels`

以下程式碼範例顯示如何使用 `DetectModerationLabels`。

如需詳細資訊，請參閱偵測不適合的映像。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun detectModLabels(sourceImage: String) {
    val myImage =
        Image {
            this.bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectModerationLabelsRequest {
            image = myImage
            minConfidence = 60f
```

```
        }

    RekognitionClient.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS適用於 Kotlin 的 SDK API 參考》中的 DetectModerationLabels。

`DetectText`

以下程式碼範例顯示如何使用 `DetectText`。

如需更多資訊，請參閱偵測映像中的文字。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun detectTextLabels(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectTextRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS適用於 Kotlin 的 SDK API 參考》中的 DetectText。

`IndexFaces`

以下程式碼範例顯示如何使用 `IndexFaces`。

如需詳細資訊，請參閱將人臉新增至集合。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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")
            }
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS適用於 Kotlin 的 SDK API 參考》中的 IndexFaces。

`ListCollections`

以下程式碼範例顯示如何使用 `ListCollections`。

如需詳細資訊，請參閱列出的集合。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listAllCollections() {
    val request =
        ListCollectionsRequest {
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS適用於 Kotlin 的 SDK API 參考》中的 ListCollections。

`ListFaces`

以下程式碼範例顯示如何使用 `ListFaces`。

如需更多資訊，請參閱集合中列出的人臉。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listFacesCollection(collectionIdVal: String?) {
    val request =
        ListFacesRequest {
            collectionId = collectionIdVal
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS適用於 Kotlin 的 SDK API 參考》中的 ListFaces。

`RecognizeCelebrities`

以下程式碼範例顯示如何使用 `RecognizeCelebrities`。

如需詳細資訊，請參閱在映像中辨識名人。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        RecognizeCelebritiesRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { 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.")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS適用於 Kotlin 的 SDK API 參考》中的 RecognizeCelebrities。

案例

建立無伺服器應用程式來管理相片

下列程式碼範例示範如何建立無伺服器應用程式，讓使用者以標籤管理相片。

SDK for Kotlin

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

偵測映像中的資訊

以下程式碼範例顯示做法：

- 啟動 Amazon Rekognition 任務，以偵測影片中的人物、物件和文字等元素。
- 檢查工作狀態，直到工作完成。

- 輸出每個工作偵測到的元素清單。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

偵測存放於 Amazon S3 儲存貯體中的人臉。

```
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.fromEnvironment { 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.fromEnvironment { 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}")
        }
    }
}
```

偵測 Amazon S3 儲存貯體中存放影片中的不當或冒犯性內容。

```
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.fromEnvironment { 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)
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS適用於 Kotlin 的 SDK API 參考》中的下列主題。
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection

- StartTextDetection

偵測映像中的物件

下列程式碼範例示範如何建置應用程式，該應用程式可使用 Amazon Rekognition 依影像中的類別偵測物件。

適用於 Kotlin 的 SDK

展示如何使用 Amazon Rekognition Kotlin API 建立應用程式，該應用程式使用 Amazon Rekognition 對位於 Amazon Simple Storage Service (Amazon S3) 儲存貯體中的映像按類別識別物件。此應用程式可使用 Amazon Simple Email Service (Amazon SES) 向管理員傳送包含結果的電子郵件通知。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Amazon Rekognition
- Amazon S3
- Amazon SES

使用適用於 Kotlin 的 SDK 的 Route 53 網域註冊範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Route 53 網域註冊，來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 基本概念

- 動作

開始使用

Hello Route 53 網域註冊

下列程式碼範例說明如何開始使用 Route 53 網域註冊。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
 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.fromEnvironment { 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}")
            }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListPrices。

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 列出目前網域和過去一年的操作。
- 檢視過去一年的帳單和網域類型對應的價格。
- 取得網域建議。
- 檢查網域的可用性和可轉移性。
- 或者，要求網域註冊。

- 取得操作詳細資訊。
- 或者，取得網域詳細資訊。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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}")
            }
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

動作

`CheckDomainAvailability`

以下程式碼範例顯示如何使用 `CheckDomainAvailability`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CheckDomainAvailability。

`CheckDomainTransferability`

以下程式碼範例顯示如何使用 `CheckDomainTransferability`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CheckDomainTransferability。

`GetDomainDetail`

以下程式碼範例顯示如何使用 `GetDomainDetail`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
```

```
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { 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}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetDomainDetail。

`GetDomainSuggestions`

以下程式碼範例顯示如何使用 `GetDomainSuggestions`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient.fromEnvironment { 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(" ")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetDomainSuggestions。

`GetOperationDetail`

以下程式碼範例顯示如何使用 `GetOperationDetail`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetOperationDetail。

`ListDomains`

以下程式碼範例顯示如何使用 `ListDomains`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listDomains() {
    Route53DomainsClient.fromEnvironment { 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}")
            }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListDomains。

`ListOperations`

以下程式碼範例顯示如何使用 `ListOperations`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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}")
            }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListOperations。

`ListPrices`

以下程式碼範例顯示如何使用 `ListPrices`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listAllPrices(domainType: String?) {
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
```

```
        }

    Route53DomainsClient.fromEnvironment { 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}")
            }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListPrices。

`RegisterDomain`

以下程式碼範例顯示如何使用 `RegisterDomain`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 RegisterDomain。

ViewBilling

以下程式碼範例顯示如何使用 ViewBilling。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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}")
            }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ViewBilling。

使用適用於 Kotlin 的 SDK 的 Amazon S3 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon S3 來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 基本概念
- 動作
- 案例

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立儲存貯體並上傳檔案到該儲存貯體。
- 從儲存貯體下載物件。
- 將物件複製至儲存貯體中的子文件夾。
- 列出儲存貯體中的物件。
- 刪除儲存貯體物件和該儲存貯體。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

動作

`CopyObject`

以下程式碼範例顯示如何使用 `CopyObject`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CopyObject。

`CreateBucket`

以下程式碼範例顯示如何使用 `CreateBucket`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createNewBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
```

```
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateBucket。

`CreateMultiRegionAccessPoint`

以下程式碼範例顯示如何使用 `CreateMultiRegionAccessPoint`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

設定 S3 控制用戶端，將請求傳送至 us-west-2 區域。

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

建立多區域存取點。

```
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"
    }
```

等待多區域存取點變成可用。

```
        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)
            }
        }
```

- 如需詳細資訊，請參閱適用於 Kotlin 的AWS SDK 開發人員指南。
- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateMultiRegionAccessPoint。

`DeleteBucketPolicy`

以下程式碼範例顯示如何使用 `DeleteBucketPolicy`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
```

```
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteBucketPolicy。

`DeleteObjects`

以下程式碼範例顯示如何使用 `DeleteObjects`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteObjects。

`GetBucketPolicy`

以下程式碼範例顯示如何使用 `GetBucketPolicy`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetBucketPolicy。

GetObject

以下程式碼範例顯示如何使用 GetObject。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getObjectBytes(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetObject。

GetObjectAcl

以下程式碼範例顯示如何使用 GetObjectAcl。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetObjectAcl。

`ListObjectsV2`

以下程式碼範例顯示如何使用 `ListObjectsV2`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { 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
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListObjectsV2。

`PutBucketAcl`

以下程式碼範例顯示如何使用 `PutBucketAcl`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 PutBucketAcl。

PutObject

以下程式碼範例顯示如何使用 PutObject。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 PutObject。

案例

建立預先簽章 URL

下列程式碼範例示範如何建立適用於 Amazon S3 預先簽署的 URL，並上傳物件。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

建立 GetObject 預先簽章的請求，並使用 URL 下載物件。

```
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
}
```

使用進階選項建立 GetObject 預先簽章的請求。

```
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
}
```

建立 PutObject 預先簽章的請求，並使用該請求上傳物件。

```
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)
}
```

- 如需詳細資訊，請參閱適用於 Kotlin 的AWS SDK 開發人員指南。

建立無伺服器應用程式來管理相片

下列程式碼範例示範如何建立無伺服器應用程式，讓使用者以標籤管理相片。

SDK for Kotlin

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

偵測映像中的物件

下列程式碼範例示範如何建置應用程式，該應用程式可使用 Amazon Rekognition 依影像中的類別偵測物件。

適用於 Kotlin 的 SDK

展示如何使用 Amazon Rekognition Kotlin API 建立應用程式，該應用程式使用 Amazon Rekognition 對位於 Amazon Simple Storage Service (Amazon S3) 儲存貯體中的映像按類別識別物件。此應用程式可使用 Amazon Simple Email Service (Amazon SES) 向管理員傳送包含結果的電子郵件通知。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Amazon Rekognition
- Amazon S3
- Amazon SES

從多區域存取點取得物件

下列程式碼範例示範如何從多區域存取點取得物件。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

將 S3 用戶端設定為使用非對稱 Sigv4 (Sigv4a) 簽署演算法。

```
        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
        }
```

使用多區域存取點 ARN (而非儲存貯體名稱) 來擷取物件。

```
    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
    }
```

- 如需詳細資訊，請參閱適用於 Kotlin 的AWS SDK 開發人員指南。
- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetObject。

使用適用於 Kotlin 的 SDK 的 SageMaker AI 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 SageMaker AI 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 動作

- 案例

開始使用

Hello SageMaker AI

下列程式碼範例示範如何開始使用 SageMaker AI。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listBooks() {
    SageMakerClient.fromEnvironment { region = "us-west-2" }.use { sageMakerClient -
>
        val response =
 sageMakerClient.listNotebookInstances(ListNotebookInstancesRequest {})
        response.notebookInstances?.forEach { item ->
            println("The notebook name is: ${item.notebookInstanceName}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListNotebookInstances。

動作

`CreatePipeline`

以下程式碼範例顯示如何使用 `CreatePipeline`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

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

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreatePipeline。

`DeletePipeline`

以下程式碼範例顯示如何使用 `DeletePipeline`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
// 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")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeletePipeline。

`DescribePipelineExecution`

以下程式碼範例顯示如何使用 `DescribePipelineExecution`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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")
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DescribePipelineExecution。

`StartPipelineExecution`

以下程式碼範例顯示如何使用 `StartPipelineExecution`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
// 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
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 StartPipelineExecution。

案例

開始使用地理空間任務和管道

以下程式碼範例顯示做法：

- 設定管道的資源。
- 設定執行地理空間任務的管道。
- 啟動管道執行。
- 監控執行的狀態。
- 檢視管道的輸出。
- 清除資源。

如需詳細資訊，請參閱在 Community.aws 上使用AWS SDKs建立和執行 SageMaker 管道。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

使用適用於 Kotlin 的 SDK 的 Secrets Manager 範例

下列程式碼範例示範如何使用AWS SDK for Kotlin 搭配 Secrets Manager 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作

動作

`GetSecretValue`

以下程式碼範例顯示如何使用 `GetSecretValue`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getValue(secretName: String?) {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient.fromEnvironment { region = "us-east-1" }.use
 { secretsClient ->
        val response = secretsClient.getSecretValue(valueRequest)
        val secret = response.secretString
        println("The secret value is $secret")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetSecretValue。

使用適用於 Kotlin 的 SDK 的 Amazon SES 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon SES 來執行動作和實作常見案例。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 案例

案例

建立 Web 應用程式以追蹤 DynamoDB 資料

下列程式碼示範如何建立一個 Web 應用程式，該應用程式會追蹤 Amazon DynamoDB 資料表中的工作項目，並使用 Amazon Simple Email Service (Amazon SES) 傳送報告。

SDK for Kotlin

說明如何使用 Amazon DynamoDB API 來建立可追蹤 DynamoDB 工作資料的動態 Web 應用程式。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- DynamoDB
- Amazon SES

建立用於追蹤 Amazon Redshift 資料的 Web 應用程式

下列程式碼範例示範如何使用 Amazon Redshift 資料庫，建立可追蹤和報告工作項目的 Web 應用程式。

SDK for Kotlin

說明如何建立可追蹤和報告存放在 Amazon Redshift 資料庫中的工作項目的 Web 應用程式。

如需完整的原始碼以及如何設定 Spring REST API 以查詢 Amazon Redshift 資料並供 React 應用程式使用的說明，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Amazon Redshift
- Amazon SES

建立 Aurora 無伺服器工作項目追蹤器

下列程式碼示範如何建立一個 Web 應用程式，該應用程式會追蹤 Amazon Aurora Serverless 資料庫中的工作項目，並使用 Amazon Simple Email Service (Amazon SES) 傳送報告。

SDK for Kotlin

說明如何建立可追蹤和報告存放在 Amazon RDS 資料庫中的工作項目的 Web 應用程式。

如需完整的原始碼以及如何設定 Spring REST API 以查詢 Amazon Aurora 無伺服器資料並供 React 應用程式使用的說明，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Aurora
- Amazon RDS
- Amazon RDS 資料服務
- Amazon SES

偵測映像中的物件

下列程式碼範例示範如何建置應用程式，該應用程式可使用 Amazon Rekognition 依影像中的類別偵測物件。

適用於 Kotlin 的 SDK

展示如何使用 Amazon Rekognition Kotlin API 建立應用程式，該應用程式使用 Amazon Rekognition 對位於 Amazon Simple Storage Service (Amazon S3) 儲存貯體中的映像按類別識別物件。此應用程式可使用 Amazon Simple Email Service (Amazon SES) 向管理員傳送包含結果的電子郵件通知。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Amazon Rekognition
- Amazon S3
- Amazon SES

使用適用於 Kotlin 的 SDK 的 Amazon SNS 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon SNS 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 動作
- 案例

開始使用

您好 Amazon SNS

下列程式碼範例示範如何開始使用 Amazon SNS。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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}")
            }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的ListTopics。

動作

`CreateTopic`

以下程式碼範例顯示如何使用 `CreateTopic`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》https://sdk.amazonaws.com/kotlin/api/latest/index.html中的 CreateTopic。

`DeleteTopic`

以下程式碼範例顯示如何使用 `DeleteTopic`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》https://sdk.amazonaws.com/kotlin/api/latest/index.html中的 DeleteTopic。

`GetTopicAttributes`

以下程式碼範例顯示如何使用 `GetTopicAttributes`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetTopicAttributes。

`ListSubscriptions`

以下程式碼範例顯示如何使用 `ListSubscriptions`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listSNSSubscriptions() {
    SnsClient.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的ListSubscriptions。

`ListTopics`

以下程式碼範例顯示如何使用 `ListTopics`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listSNSTopics() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的ListTopics。

`Publish`

以下程式碼範例顯示如何使用 `Publish`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun pubTopic(
    topicArnVal: String,
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的發佈。

`SetTopicAttributes`

以下程式碼範例顯示如何使用 `SetTopicAttributes`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun setTopAttr(
```

```
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》https://sdk.amazonaws.com/kotlin/api/latest/index.html中的 SetTopicAttributes。

`Subscribe`

以下程式碼範例顯示如何使用 `Subscribe`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

透過電子郵件地址訂閱某個主題。

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
        SubscribeRequest {
```

```
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

將 Lambda 函式訂閱至主題。

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的訂閱。

`TagResource`

以下程式碼範例顯示如何使用 `TagResource`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 TagResource。

`Unsubscribe`

以下程式碼範例顯示如何使用 Unsubscribe。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的取消訂閱。

案例

建置 Amazon SNS 應用程式

以下程式碼範例示範如何建立具有訂閱和發布功能，並且可轉譯訊息的應用程式。

SDK for Kotlin

示範如何使用 Amazon SNS Kotlin API 來建立具有訂閱和發布功能的應用程式。此外，此範例應用程式也會轉譯訊息。

如需完整的原始碼和如何建立 Web 應用程式的指示，請參閱 GitHub 上的完整範例。

如需完整的原始碼和如何建立原生 Android 應用程式的指示，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Amazon SNS
- Amazon Translate

建立無伺服器應用程式來管理相片

下列程式碼範例示範如何建立無伺服器應用程式，讓使用者以標籤管理相片。

SDK for Kotlin

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

發布簡訊

下列程式碼範例示範如何使用 Amazon SNS 發佈訊息。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun pubTextSMS(
```

```
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的發佈。

將訊息發佈至佇列

以下程式碼範例顯示做法：

- 建立主題 (FIFO 或非 FIFO)。
- 為主題訂閱多個佇列，並提供套用篩選條件的選擇。
- 發佈訊息至主題。
- 輪詢佇列以獲取收到的訊息。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - CreateQueue
  - CreateTopic

- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- 發布
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

使用適用於 Kotlin 的 SDK 的 Amazon SQS 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon SQS 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 動作
- 案例

開始使用

Hello Amazon SQS

下列程式碼範例說明如何開始使用 Amazon SQS。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient
            .listQueuesPaginated { }
            .transform { it.queueUrls?.forEach { queue -> emit(queue) } }
            .collect { queue ->
                println("The Queue URL is $queue")
            }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListQueues。

動作

`CreateQueue`

以下程式碼範例顯示如何使用 `CreateQueue`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createQueue(queueNameVal: String): String {
    println("Create Queue")
    val createQueueRequest =
        CreateQueueRequest {
            queueName = queueNameVal
        }

    SqsClient.fromEnvironment { 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()
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateQueue。

`DeleteMessage`

以下程式碼範例顯示如何使用 `DeleteMessage`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteMessage。

`DeleteQueue`

以下程式碼範例顯示如何使用 `DeleteQueue`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteQueue。

`ListQueues`

以下程式碼範例顯示如何使用 `ListQueues`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listQueues() {
    println("\nList Queues")

    val prefix = "que"
    val listQueuesRequest =
        ListQueuesRequest {
            queueNamePrefix = prefix
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.listQueues(listQueuesRequest)
        response.queueUrls?.forEach { url ->
            println(url)
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListQueues。

`ReceiveMessage`

以下程式碼範例顯示如何使用 `ReceiveMessage`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun receiveMessages(queueUrlVal: String?) {
    println("Retrieving messages from $queueUrlVal")

    val receiveMessageRequest =
        ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.receiveMessage(receiveMessageRequest)
        response.messages?.forEach { message ->
            println(message.body)
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ReceiveMessage。

SendMessage

以下程式碼範例顯示如何使用 SendMessage。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun sendMessages(
    queueUrlVal: String,
    message: String,
) {
    println("Sending multiple messages")
    println("\nSend message")
    val sendRequest =
        SendMessageRequest {
            queueUrl = queueUrlVal
```

```
            messageBody = message
            delaySeconds = 10
        }

    SqsClient.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessageBatch(sendMessageBatchRequest)
        println("Batch message were successfully sent.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 SendMessage。

案例

建立即時通訊軟體

下列程式碼範例示範如何搭配使用 Amazon SQS 建立即時通訊軟體。

SDK for Kotlin

示範如何使用 Amazon SQS API 開發用於傳送和擷取訊息的 Spring REST API。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Amazon Comprehend
- Amazon SQS

將訊息發佈至佇列

以下程式碼範例顯示做法：

- 建立主題 (FIFO 或非 FIFO)。
- 為主題訂閱多個佇列，並提供套用篩選條件的選擇。
- 發佈訊息至主題。
- 輪詢佇列以獲取收到的訊息。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - CreateQueue
  - CreateTopic

- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- 發布
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

使用適用於 Kotlin 的 SDK 的 Step Functions 範例

下列程式碼範例示範如何使用AWS SDK for Kotlin 搭配 Step Functions 來執行動作和實作常見案例。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 基本概念
- 動作

開始使用

Hello Step Functions

下列程式碼範例示範如何開始使用 Step Functions。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》的 ListStateMachines。

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 建立活動。
- 從 Amazon States Language 定義建立狀態機器，其中包含先前建立的活動步驟。
- 執行狀態機器，並使用使用者輸入回應活動。
- 在執行完成後取得最終狀態和輸出，然後清除資源。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines
  - SendTaskSuccess
  - StartExecution
  - StopExecution

動作

`CreateActivity`

以下程式碼範例顯示如何使用 `CreateActivity`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createActivity(activityName: String): String? {
```

```
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateActivity。

`CreateStateMachine`

以下程式碼範例顯示如何使用 `CreateStateMachine`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
```

```
        return response.stateMachineArn
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateStateMachine。

`DeleteActivity`

以下程式碼範例顯示如何使用 `DeleteActivity`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteActivity。

`DeleteStateMachine`

以下程式碼範例顯示如何使用 `DeleteStateMachine`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DeleteStateMachine。

`DescribeExecution`

以下程式碼範例顯示如何使用 `DescribeExecution`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }
```

```
    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient.fromEnvironment { 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")
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DescribeExecution。

`DescribeStateMachine`

以下程式碼範例顯示如何使用 `DescribeStateMachine`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
```

```
    SfnClient.fromEnvironment { 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}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 DescribeStateMachine。

`GetActivityTask`

以下程式碼範例顯示如何使用 `GetActivityTask`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 GetActivityTask。

`ListActivities`

以下程式碼範例顯示如何使用 `ListActivities`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listAllActivites() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }

    SfnClient.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListActivities。

`ListExecutions`

以下程式碼範例顯示如何使用 `ListExecutions`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getExeHistory(exeARN: String?) {
    val historyRequest =
        GetExecutionHistoryRequest {
            executionArn = exeARN
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getExecutionHistory(historyRequest)
        response.events?.forEach { event ->
            println("The event type is ${event.type}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ListExecutions。

`ListStateMachines`

以下程式碼範例顯示如何使用 `ListStateMachines`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》的 ListStateMachines。

SendTaskSuccess

以下程式碼範例顯示如何使用 SendTaskSuccess。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
            output = json
```

```
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 SendTaskSuccess。

`StartExecution`

以下程式碼範例顯示如何使用 `StartExecution`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 StartExecution。

支援使用適用於 Kotlin 的 SDK 的範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 來執行動作和實作常見案例支援。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 開始使用
- 基本概念
- 動作

開始使用

您好支援

下列程式碼範例示範如何開始使用支援。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
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.fromEnvironment { 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++
            }
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeServices。

基本概念

了解基本概念

以下程式碼範例顯示做法：

- 取得並顯示案例可用的服務和嚴重性層級。
- 根據選取的服務、類別和嚴重性層級建立支援案例。
- 取得並顯示當天開啟的案例清單。
- 將附件集和通訊新增至新案例。
- 描述案例的新附件和通訊。
- 解決案例。
- 取得並顯示當天已解決的案例清單。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
/**
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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

動作

`AddAttachmentsToSet`

以下程式碼範例顯示如何使用 `AddAttachmentsToSet`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 AddAttachmentsToSet。

`AddCommunicationToCase`

以下程式碼範例顯示如何使用 `AddCommunicationToCase`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient.fromEnvironment { 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")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 AddCommunicationToCase。

`CreateCase`

以下程式碼範例顯示如何使用 `CreateCase`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 CreateCase。

`DescribeAttachment`

以下程式碼範例顯示如何使用 `DescribeAttachment`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeAttachment。

`DescribeCases`

以下程式碼範例顯示如何使用 `DescribeCases`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
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.fromEnvironment { 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}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeCases。

`DescribeCommunications`

以下程式碼範例顯示如何使用 `DescribeCommunications`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
            caseId = caseIdVal
            maxResults = 10
        }

    SupportClient.fromEnvironment { 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 ""
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeCommunications。

`DescribeServices`

以下程式碼範例顯示如何使用 `DescribeServices`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
// 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.fromEnvironment { 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
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeServices。

`DescribeSeverityLevels`

以下程式碼範例顯示如何使用 `DescribeSeverityLevels`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun displaySevLevels(): String {
```

```
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { 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
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 DescribeSeverityLevels。

`ResolveCase`

以下程式碼範例顯示如何使用 `ResolveCase`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
```

```
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的AWS SDK API 參考》中的 ResolveCase。

使用適用於 Kotlin 的 SDK 的 Amazon Translate 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 Amazon Translate 來執行動作和實作常見案例。

案例是向您展示如何呼叫服務中的多個函數或與其他AWS 服務組合來完成特定任務的程式碼範例。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 案例

案例

建置 Amazon SNS 應用程式

以下程式碼範例示範如何建立具有訂閱和發布功能，並且可轉譯訊息的應用程式。

SDK for Kotlin

示範如何使用 Amazon SNS Kotlin API 來建立具有訂閱和發布功能的應用程式。此外，此範例應用程式也會轉譯訊息。

如需完整的原始碼和如何建立 Web 應用程式的指示，請參閱 GitHub 上的完整範例。

如需完整的原始碼和如何建立原生 Android 應用程式的指示，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Amazon SNS
- Amazon Translate

使用適用於 Kotlin 的 SDK 的 X-Ray 範例

下列程式碼範例示範如何使用適用於 Kotlin 的AWS SDK 搭配 X-Ray 來執行動作和實作常見案例。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

每個範例均包含完整原始碼的連結，您可在連結中找到如何設定和執行內容中程式碼的相關指示。

主題

- 動作

動作

`CreateGroup`

以下程式碼範例顯示如何使用 `CreateGroup`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createNewGroup(groupNameVal: String?) {
    val groupRequest =
        CreateGroupRequest {
            filterExpression = "fault = true AND http.url CONTAINS \"example/game\"
 AND responsetime >= 5"
            groupName = groupNameVal
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val groupResponse = xRayClient.createGroup(groupRequest)
        println("The Group ARN is " + (groupResponse.group?.groupArn))
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Kotlin 的 SDK API 參考》中的 CreateGroup。

`CreateSamplingRule`

以下程式碼範例顯示如何使用 `CreateSamplingRule`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun createRule(ruleNameVal: String?) {
    val rule =
        SamplingRule {
            ruleName = ruleNameVal
            priority = 1
            httpMethod = "*"
            serviceType = "*"
            serviceName = "*"
            urlPath = "*"
            version = 1
            host = "*"
            resourceArn = "*"
        }

    val ruleRequest =
        CreateSamplingRuleRequest {
            samplingRule = rule
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val ruleResponse: CreateSamplingRuleResponse =
 xRayClient.createSamplingRule(ruleRequest)
        println("The ARN of the new rule is
 ${ruleResponse.samplingRuleRecord?.samplingRule?.ruleArn}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Kotlin 的 SDK API 參考》中的 CreateSamplingRule。

DeleteGroup

以下程式碼範例顯示如何使用 DeleteGroup。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteSpecificGroup(groupNameVal: String) {
    val groupRequest =
        DeleteGroupRequest {
            groupName = groupNameVal
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        xRayClient.deleteGroup(groupRequest)
        println("$groupNameVal was deleted!")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Kotlin 的 SDK API 參考》中的 DeleteGroup。

DeleteSamplingRule

以下程式碼範例顯示如何使用 DeleteSamplingRule。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun deleteRule(ruleNameVal: String?) {
    val ruleRequest =
        DeleteSamplingRuleRequest {
            ruleName = ruleNameVal
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        xRayClient.deleteSamplingRule(ruleRequest)
        println("$ruleNameVal was deleted")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Kotlin 的 SDK API 參考》中的 DeleteSamplingRule。

`GetGroups`

以下程式碼範例顯示如何使用 `GetGroups`。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getAllGroups() {
    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val response = xRayClient.getGroups(GetGroupsRequest {})
```

```
        response.groups?.forEach { group ->
            println("The AWS X-Ray group name is ${group.groupName}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Kotlin 的 SDK API 參考》中的 GetGroups。

GetSamplingRules

以下程式碼範例顯示如何使用 GetSamplingRules。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getRules() {
    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val response = xRayClient.getSamplingRules(GetSamplingRulesRequest {})
        response.samplingRuleRecords?.forEach { record ->
            println("The rule name is ${record.samplingRule?.ruleName}")
            println("The related service is: ${record.samplingRule?.serviceName}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Kotlin 的 SDK API 參考》中的 GetSamplingRules。

GetServiceGraph

以下程式碼範例顯示如何使用 GetServiceGraph。

適用於 Kotlin 的 SDK

> **Note**
>
> GitHub 上提供更多範例。尋找完整範例，並了解如何在 AWS程式碼範例儲存庫中設定和執行。

```
suspend fun getGraph(groupNameVal: String?) {
    val time = aws.smithy.kotlin.runtime.time.Instant
    val getServiceGraphRequest =
        GetServiceGraphRequest {
            groupName = groupNameVal
            this.startTime = time.now()
            endTime = time.now()
        }
    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val response = xRayClient.getServiceGraph(getServiceGraphRequest)
        response.services?.forEach { service ->
            println("The name of the service is  ${service.name}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Kotlin 的 SDK API 參考》中的 GetServiceGraph。

的安全性適用於 Kotlin 的 AWS SDK

雲端安全是 Amazon Web Services (AWS) 最重視的一環。身為AWS客戶的您，將能從資料中心和網路架構的建置中獲益，以滿足組織最為敏感的安全要求。安全性是AWS與您之間的共同責任。共同責任模型 將此描述為雲端本身的安全和雲端內部的安全。

雲端的安全性 –AWS負責保護執行AWS雲端中提供的所有服務的基礎設施，並為您提供可安全使用的服務。我們的安全責任是 的最高優先順序AWS，我們的安全有效性由第三方稽核人員定期測試和驗證，作為AWS合規計劃的一部分。

雲端的安全性 – 您的責任取決於您使用AWS的服務，以及其他因素，包括資料的敏感度、組織的需求，以及適用的法律和法規。

此AWS產品或服務會透過其支援的特定 Amazon Web Services (AWS) 服務，遵循共同責任模型。如需AWS服務安全資訊，請參閱AWS服務安全文件頁面，以及AWS合規計劃在AWS合規工作範圍內的服務。

主題

- 中的資料保護適用於 Kotlin 的 AWS SDK
- 適用於 Kotlin 的 AWS SDK支援 Transport Layer Security (TLS)
- 身分和存取權管理
- 此AWS產品或服務的合規驗證
- 此AWS產品或服務的彈性
- 此AWS產品或服務的基礎設施安全

中的資料保護適用於 Kotlin 的 AWS SDK

將AWS共同責任模式應用於適用於 Kotlin 的 AWS SDK中的資料保護。如此模型所述，AWS負責保護執行所有 的全域基礎設施AWS 雲端。您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用AWS 服務的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需歐洲資料保護的詳細資訊，請參閱一般資料保護規則 (GDPR) 中心。

基於資料保護目的，我們建議您保護AWS 帳戶登入資料，並使用AWS IAM Identity Center或AWS Identity and Access Management(IAM) 設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。

- 使用 SSL/TLS 與AWS資源通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用 設定 API 和使用者活動記錄AWS CloudTrail。如需有關使用 CloudTrail 追蹤擷取AWS活動的資訊，請參閱AWS CloudTrail《 使用者指南》中的使用 CloudTrail 追蹤。
- 使用AWS加密解決方案，以及其中的所有預設安全控制AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在AWS透過命令列界面或 API 存取 時需要 FIPS 140-3 驗證的密碼編譯模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-3。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用適用於 Kotlin 的 SDK 或使用主控台、API AWS CLI或AWS SDKs 的其他AWS 服務時。您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

適用於 Kotlin 的 AWS SDK支援 Transport Layer Security (TLS)

以下資訊僅適用於 Java SSL 實作 (中以 JVM 為適用於 Kotlin 的 AWS SDK目標的預設 SSL 實作)。如果您使用不同的 SSL 實作，請參閱特定的 SSL 實作，以了解如何強制執行 TLS 版本。

Java 中的 TLS 支援

若要使用AWS服務，基礎 JDK 必須支援最低版本的 TLS 1.2，但建議使用 TLS 1.3。TLS 1.3 是啟用量子後密碼編譯的先決條件，可能需要額外的動作或組態。若要進一步了解，請參閱啟用混合式後量子 TLS。

從 Java 7 開始支援 TLS 1.2。從 Java 11 開始支援 TLS 1.3，雖然有些 JVM 供應商已將支援回溯到舊版 Java。

如何檢查 TLS 版本

若要檢查 Java 虛擬機器 (JVM) 中支援哪些 TLS 版本，您可以使用下列程式碼。

```
println(SSLContext.getDefault().supportedSSLParameters.protocols.joinToString(separator
 = ", "))
```

要查看運作中的 SSL 交握和使用的 TLS 版本，您可以使用系統屬性 javax.net.debug。

```
-Djavax.net.debug=ssl
```

身分和存取權管理

AWS Identity and Access Management(IAM) 是AWS 服務，可協助管理員安全地控制對AWS資源的存取。IAM 管理員可控制誰可以進行驗證 （ 登入 ） 和授權 （ 具有許可 ） 來使用AWS資源。IAM 是您可以免費使用AWS 服務的 。

主題

- 目標對象
- 使用身分驗證
- 使用政策管理存取權
- AWS 服務如何使用 IAM
- 對AWS身分和存取進行故障診斷

目標對象

您使用AWS Identity and Access Management(IAM) 的方式會有所不同，取決於您在 中執行的工作AWS。

服務使用者 – 如果您使用AWS 服務執行任務，管理員會為您提供所需的登入資料和許可。當您使用更多AWS功能來執行工作時，您可能需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 中的功能AWS，請參閱 對AWS身分和存取進行故障診斷或AWS 服務您正在使用的 使用者指南。

服務管理員 – 如果您負責公司AWS的資源，您可能擁有 的完整存取權AWS。您的任務是判斷服務使用者應存取哪些AWS功能和資源。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司如何使用 IAM AWS，請參閱您正在使用的 使用者指南AWS 服務。

IAM 管理員：如果您是 IAM 管理員，建議您掌握如何撰寫政策以管理AWS存取權的詳細資訊。若要檢視您可以在 IAM 中使用的以AWS身分為基礎的政策範例，請參閱AWS 服務您正在使用的 使用者指南。

使用身分驗證

身分驗證是您AWS使用身分憑證登入 的方式。您必須以AWS 帳戶根使用者、IAM 使用者或擔任 IAM 角色身分進行身分驗證。

您可以使用身分來源的登入資料，例如AWS IAM Identity Center(IAM Identity Center)、單一登入身分驗證或 Google/Facebook 登入資料，以聯合身分的形式登入。如需有關登入的詳細資訊，請參閱《AWS 登入使用者指南》中的如何登入您的AWS 帳戶。

對於程式設計存取，AWS提供 SDK 和 CLI 以密碼編譯方式簽署請求。如需詳細資訊，請參閱《IAM 使用者指南》中的 API 請求的AWS第 4 版簽署程序。

AWS 帳戶根使用者

當您建立 時AWS 帳戶，您會從一個名為AWS 帳戶theroot 使用者的登入身分開始，該身分具有對所有 AWS 服務和 資源的完整存取權。強烈建議不要使用根使用者來執行日常任務。有關需要根使用者憑證的任務，請參閱《IAM 使用者指南》中的需要根使用者憑證的任務。

聯合身分

最佳實務是要求人類使用者使用聯合身分提供者，以AWS 服務使用臨時憑證存取 。

聯合身分是您企業目錄、Web 身分提供者的使用者，或使用來自身分來源的AWS 服務憑證存取 Directory Service。聯合身分會擔任角色，而該角色會提供臨時憑證。

若需集中化管理存取權限，建議使用AWS IAM Identity Center。如需詳細資訊，請參閱 AWS IAM Identity Center使用者指南中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html是一種身分具備單人或應用程式的特定許可權。建議以臨時憑證取代具備長期憑證的 IAM 使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的要求人類使用者使用聯合身分提供者來AWS使用臨時憑證存取 。

IAM 群組會指定 IAM 使用者集合，使管理大量使用者的許可權更加輕鬆。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 使用者的使用案例。

IAM 角色

IAM 角色https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html的身分具有特定許可權，其可以提供臨時憑證。您可以透過從使用者切換到 IAM 角色（主控台）或呼叫AWS CLI或AWS API 操作來擔任角色。如需詳細資訊，請參閱《IAM 使用者指南》中的擔任角色的方法。

IAM 角色適用於聯合身分使用者存取、臨時 IAM 使用者許可、跨帳戶存取權與跨服務存取，以及在 Amazon EC2 執行的應用程式。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 中的快帳戶資源存取。

使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源AWS來控制AWS中的存取。政策定義與身分或資源相關聯的許可。當委託人提出請求時AWS， 會評估這些政策。大多數政策會以 JSON 文件AWS的形式存放在 中。如需進一步了解 JSON 政策文件，請參閱《IAM 使用者指南》中的 JSON 政策概觀。

管理員會使用政策，透過定義哪些主體可在哪些條件下對哪些資源執行動作，以指定可存取的範圍。

預設情況下，使用者和角色沒有許可。IAM 管理員會建立 IAM 政策並將其新增至角色，供使用者後續擔任。IAM 政策定義動作的許可，無論採用何種方式執行。

身分型政策

身分型政策是附加至身分 (使用者、使用者群組或角色) 的 JSON 許可政策文件。這類政策控制身分可對哪些資源執行哪些動作，以及適用的條件。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

身分型政策可分為內嵌政策 (直接內嵌於單一身分) 與受管政策 (可附加至多個身分的獨立政策)。如需了解如何在受管政策及內嵌政策之間做選擇，請參閱《IAM 使用者指南》中的在受管政策與內嵌政策之間選擇。

資源型政策

資源型政策是附加到資源的 JSON 政策文件。範例包括 IAM 角色信任政策與 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。您必須在資源型政策中指定主體。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 IAM 的AWS受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 AWS WAF和 Amazon VPC 是支援 ACLs的服務範例。如需進一步了解 ACL，請參閱《Amazon Simple Storage Service 開發人員指南》中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS支援其他政策類型，可設定更多常見政策類型授予的最大許可：

- 許可界限 — 設定身分型政策可授與 IAM 實體的最大許可。如需詳細資訊，請參閱《 IAM 使用者指南》中的 IAM 實體許可界限。
- 服務控制政策 (SCP) — 為AWS Organizations中的組織或組織單位指定最大許可。如需詳細資訊，請參閱《AWS Organizations使用者指南》中的服務控制政策。
- 資源控制政策 (RCP) — 設定您帳戶中資源可用許可的上限。如需詳細資訊，請參閱《AWS Organizations使用者指南》中的資源控制政策 (RCP)。
- 工作階段政策 — 在以程式設計方式為角色或聯合身分使用者建立臨時工作階段時，以參數形式傳遞的進階政策。如需詳細資訊，請參《IAM 使用者指南》中的工作階段政策。

多種政策類型

當多種類型的政策適用於請求時，產生的許可會更複雜而無法理解。若要了解如何AWS在涉及多種政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

AWS 服務如何使用 IAM

若要深入了解 如何使用AWS 服務大多數 IAM 功能，請參閱《IAM 使用者指南》中的AWS使用 IAM 的服務。

若要了解如何AWS 服務搭配 IAM 使用特定 ，請參閱相關服務使用者指南的安全章節。

對AWS身分和存取進行故障診斷

使用以下資訊來協助您診斷和修正使用AWS和 IAM 時可能遇到的常見問題。

主題

- 我無權在 中執行動作AWS
- 我未獲得執行 iam:PassRole 的授權
- 我想要允許 以外的人員AWS 帳戶存取我的AWS資源

我無權在 中執行動作AWS

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在`mateojackson` IAM 使用者嘗試使用主控台檢視一個虛構 *`my-example-widget`* 資源的詳細資訊，但卻無虛構 `awes:`*`GetWidget`* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 mateojackson 使用者的政策，允許使用 awes:*GetWidget* 動作存取 *my-example-widget* 資源。

如果您需要協助，請聯絡您的AWS管理員。您的管理員提供您的簽署憑證。

我未獲得執行 iam:PassRole 的授權

如果您收到錯誤，告知您未獲授權執行 iam:PassRole 動作，您的政策必須更新，允許您將角色傳遞給AWS。

有些AWS 服務可讓您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

名為 marymajor 的 IAM 使用者嘗試使用主控台在AWS中執行動作時，發生下列範例錯誤。但是，動作要求服務具備服務角色授予的許可。Mary 沒有將角色傳遞給服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 iam:PassRole 動作。

如果您需要協助，請聯絡您的AWS管理員。您的管理員提供您的簽署憑證。

我想要允許 以外的人員AWS 帳戶存取我的AWS資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 是否AWS支援這些功能，請參閱 AWS 服務如何使用 IAM。
- 若要了解如何AWS 帳戶在您擁有的 資源之間提供存取權，請參閱《IAM 使用者指南》中的在您擁有 AWS 帳戶的另一個 IAM 使用者中提供存取權。
- 若要了解如何將資源的存取權提供給第三方AWS 帳戶，請參閱《IAM 使用者指南》中的將存取權提供給第三方AWS 帳戶擁有。

- 如需了解如何透過聯合身分提供存取權，請參閱《IAM 使用者指南》中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 中的跨帳戶資源存取。

此AWS產品或服務的合規驗證

若要了解AWS 服務是否在特定合規計劃範圍內，請參閱AWS 服務合規計劃範圍內然後選擇您感興趣的合規計劃。如需一般資訊，請參閱AWS合規計劃。

您可以使用 下載第三方稽核報告AWS Artifact。如需詳細資訊，請參閱下載報告 in AWS Artifact

您使用 時的合規責任AWS 服務取決於資料的機密性、您公司的合規目標，以及適用的法律和法規。如需使用 時合規責任的詳細資訊AWS 服務，請參閱 AWS安全文件。

此AWS產品或服務會透過其支援的特定 Amazon Web Services (AWS) 服務，遵循共同責任模型。如需AWS服務安全資訊，請參閱AWS服務安全文件頁面，以及AWS合規計劃在AWS合規工作範圍內的服務。

此AWS產品或服務的彈性

AWS全球基礎設施是以AWS 區域和可用區域為基礎建置。

AWS 區域提供多個實體分隔和隔離的可用區域，這些可用區域與低延遲、高輸送量和高備援聯網連接。

透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需AWS區域和可用區域的詳細資訊，請參閱 AWS全球基礎設施。

此AWS產品或服務會透過其支援的特定 Amazon Web Services (AWS) 服務，遵循共同責任模型。如需AWS服務安全資訊，請參閱AWS服務安全文件頁面，以及AWS合規計劃在AWS合規工作範圍內的服務。

此AWS產品或服務的基礎設施安全

此AWS產品或服務使用 受管服務，因此受到 全球網路安全的AWS保護。如需AWS安全服務以及如何AWS保護基礎設施的資訊，請參閱AWS雲端安全。若要使用基礎設施安全的最佳實務來設計您的AWS環境，請參閱安全支柱AWS Well-Architected Framework 中的基礎設施保護。

您可以使用AWS發佈的 API 呼叫，透過網路存取此AWS產品或服務。使用者端必須支援下列專案：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

此AWS產品或服務會透過其支援的特定 Amazon Web Services (AWS) 服務，遵循共同責任模型。如需AWS服務安全資訊，請參閱AWS服務安全文件頁面，以及AWS合規計劃在AWS合規工作範圍內的服務。

文件歷史紀錄

本主題說明 適用於 Kotlin 的 AWS SDK 開發人員指南在其歷史記錄過程中的重要變更。

變更	描述	日期
the section called “重試”	重試內容已重新組織，並已新增可重試例外狀況的詳細資訊。	2025 年 9 月 10 日
新增如何快取獨立登入資料提供者登入資料的相關資訊	使用獨立提供者快取登入資料	2025 年 6 月 17 日
the section called “模擬”	新增使用適用於 Kotlin 的 SDK 模擬和使用 MockK 的相關資訊	2025 年 4 月 30 日
新增如何使用 Gradle 解決與 SDK 之相依性衝突的相關資訊	如何解決相依性衝突？	2025 年 4 月 15 日
使用檢查總和保護資料完整性	內容已更新，其中包含自動檢查總和計算的詳細資訊。	2025 年 1 月 16 日
更新預設登入資料提供者鏈結內容	預設登入資料提供者鏈結。	2025 年 1 月 15 日
更新建置檔案範例	顯示 Gradle 8.11.1 版產生的建置檔案元素。顯示 BOM 的使用。內嵌連結至最新版本的成品。	2024 年 12 月 18 日
新增 DynamoDB Mapper （開發人員預覽） 主題	使用 DynamoDB Mapper 將類別映射至 DynamoDB 項目（開發人員預覽版）	2024 年 10 月 29 日
更新 Amazon S3 儲存貯體名稱	使用 的 Amazon S3 檢查總和適用於 Kotlin 的 AWS SDK	2024 年 9 月 30 日
新增 OkHttp4 引擎的資訊	指定 HTTP 引擎類型	2024 年 9 月 26 日

新增 DynamoDB AWS 帳戶型端點的相關資訊	使用以 AWS 帳戶為基礎的端點	2024 年 9 月 24 日
新增疑難排解FAQs主題	疑難排解常見問答集	2024 年 9 月 18 日
更新 OpenTelemetry 組態範例和預設全域遙測供應商的組態	可觀測性	2024 年 5 月 2 日
提供有關服務用戶端建立程序的詳細資訊	建立服務用戶端	2024 年 3 月 14 日
新增多區域存取點主題	使用適用於 Kotlin 的 SDK 來使用 Amazon S3 多區域存取點	2024 年 2 月 6 日
新增 Gradle 版本目錄指示	Gradle 版本目錄 （標籤）	2023 年 12 月 19 日
一般可用性版本	適用於 Kotlin 的 AWS SDK 開發人員指南	2023 年 11 月 27 日
根據 SDK 更新更新用戶端端點組態區段	用戶端端點	2023 年 8 月 25 日
Amazon S3 檢查總和	新增章節，說明如何搭配 Amazon S3 使用彈性檢查總和。	2023 年 8 月 14 日
新增可觀測性主題	可觀測性	2023 年 8 月 3 日
新增討論重試的主題	重試	2023 年 7 月 7 日
根據 SDK 更新更新 HTTP 用戶端組態區段	HTTP 用戶端組態	2023 年 6 月 6 日
新增 HTTP 預先簽署主題	預先簽署請求	2023 年 6 月 2 日
新增 HTTP 攔截器主題	HTTP 攔截器	2023 年 5 月 22 日
支援自動權杖重新整理	更新單一登入存取的指示。	2023 年 5 月 18 日
Amazon S3 檢查總和	新增章節，說明如何搭配 Amazon S3 使用檢查總和。	2023 年 5 月 15 日

覆寫服務用戶端組態	新增章節，說明如何覆寫服務用戶端的組態，並說明資源如何受到影響。	2023 年 5 月 8 日
強制執行最低 TLS 版本	新增章節，說明強制執行最低 TLS 版本的選項。	2023 年 5 月 3 日
用戶端端點組態	新增討論用戶端端點組態的主題。	2023 年 4 月 7 日
IAM 最佳實務更新	更新了指南以符合 IAM 最佳實務。如需更多詳細資訊，請參閱 IAM 中的安全最佳實務。	2023 年 3 月 22 日
新增範例 Maven 專案檔案	除了設定主題中的 Gradle 中的專案檔案之外，還會顯示 Maven 專案檔案的範例。	2022 年 12 月 2 日
修訂 開發人員指南預覽版的內容	已更新內容以反映最近的開發工作	2022 年 10 月 4 日
適用於 Kotlin 的 AWS SDK 開發人員預覽版	適用於 Kotlin 的 AWS SDK	2021 年 12 月 2 日
適用於 Kotlin 的 AWS SDK alpha 版本	宣布新的 適用於 Kotlin 的 AWS SDK Alpha 版本	2021 年 8 月 30 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。