개발자 안내서

AWS SDK for Kotlin

AWS SDK for Kotlin: 개발자 안내서

Table of Contents

란 무엇입니까 AWS SDK for Kotlin? 1
SDK 시작 1
SDK 메이저 버전에 대한 유지 관리 및 지원 1
추가 리소스 1
시작 3
1단계: 튜토리얼 설정 3
2단계: 프로젝트 생성 3
3단계: 코드 작성 6
4단계: 애플리케이션 빌드 및 실행 8
Success 8
정리 9
다음 단계 9
설정 10
기본 설정 10
개요 10
AWS 액세스 포털에 로그인 기능 11
Single Sign-On 구성 12
를 사용하여 로그인 AWS CLI 12
Java 및 빌드 도구 설치 13
임시 자격 증명 사용 14
프로젝트 빌드 파일 생성 15
프로젝트 코딩 20
를 사용하여 로그인 AWS CLI 20
구성 22
서비스 클라이언트 생성 24
코드로 클라이언트 구성 24
환경에서 클라이언트 구성 25
클라이언트 닫기 26
AWS 리전 선택 26
기본 리전 공급자 체인 27
보안 인증 제공업체 27
기본 자격 증명 공급자 체인 사용 28
명시적 자격 증명 공급자 30
클라이언트 엔드포인트 31

사용자 지정 구성 31
예시 34
HTTP 35
HTTP 클라이언트 구성 36
HTTP 프록시 사용 39
인터셉터 39
최소 TLS 버전 적용 41
재시도 42
기본 구성 42
최대 시도 횟수 43
지연 및 백오프 43
토큰 버킷 재시도 46
적응형 재시도 48
관찰성 49
구성 TelemetryProvider 49
Metrics 51
로깅 54
원격 측정 공급자 57
클라이언트 구성 재정의 58
재정의된 클라이언트 수명 주기 59
공유 리소스 59
SDK 사용 61
요청을 생성 61
서비스 인터페이스 DSL 과부하 62
필수 입력이 없는 요청 63
코루틴 63
동시 요청 63
차단 요청 64
스트리밍 작업 65
스트리밍 응답 65
스트리밍 요청 66
페이지 매김 67
Waiters 68
오류 처리 69
서비스 예외 69
클라이언트 예외 69

오류 메타데이터 69
사전 서명 요청 70
기본 사항 사전 지정 70
고급 사전 서명 구성 71
사전 서명 POST 및 PUT 요청 72
가 미리 서명SDK할 수 있는 작업 73
FAQs 문제 해결 73
“연결 종료” 문제를 해결하려면 어떻게 해야 합니까? 73
최대 시도 횟수에 도달하기 전에 예외가 발생하는 이유는 무엇입니까? 74
NoSuchMethodError 또는를 수정하려면 어떻게 해야 합니까 NoClassDefFoundError? 75
작업 AWS 서비스 77
Amazon S3 78
체크섬을 통한 데이터 무결성 보호 79
다중 리전 액세스 포인트 작업 83
DynamoDB 88
AWS 계정 기반 엔드포인트 사용 88
DynamoDB Mapper 사용(개발자 미리 보기) 89
코드 예제 115
API Gateway 116
시나리오 117
Aurora 117
기본 사항 118
작업 130
시나리오 117
오토 스케일링 145
기본 사항 118
작업 130
Amazon Bedrock 162
작업 130
CloudWatch 163
기본 사항 118
작업 130
CloudWatch Logs 203
작업 130
Amazon Cognito 자격 증명 공급자 207
작업 130

시나리오 117
Amazon Comprehend 222
시나리오 117
DynamoDB 223
기본 사항 118
작업 130
시나리오 117
Amazon EC2 253
기본 사항 118
작업 130
Amazon ECR 283
기본 사항 118
작업 130
OpenSearch Service 313
작업 130
EventBridge 316
기본 사항 118
작업 130
AWS Glue 347
기본 사항 118
작업 130
IAM 358
기본 사항 118
작업 130
AWS IoT 377
기본 사항 118
작업 130
AWS IoT data 401
작업 130
Amazon Keyspaces 403
기본 사항 118
작업 130
AWS KMS 428
작업 130
Lambda 437
기본 사항 118

작업 130
시나리오 117
MediaConvert 446
작업 130
Amazon Pinpoint 460
작업 130
Amazon RDS 470
기본 사항 118
작업 130
시나리오 117
Amazon RDS 488
시나리오 117
Amazon Redshift 489
작업 130
시나리오 117
Amazon Rekognition 493
작업 130
시나리오 117
Route 53 도메인 등록 512
기본 사항 118
작업 130
Amazon S3 531
기본 사항 118
작업 130
시나리오 117
SageMaker AI 552
작업 130
시나리오 117
Secrets Manager 577
작업 130
Amazon SES 579
시나리오 117
Amazon SNS 581
작업 130
시나리오 117
Amazon SQS 608

작업 130
시나리오 117
Step Functions 630
기본 사항 118
작업 130
지원 651
기본 사항 118
작업 130
Amazon Translate 669
시나리오 117
보안 670
데이터 보호 670
TLS 1.2 적용 671
Java에서의 TLS 지원 671
TLS 버전을 확인하는 방법 671
ID 및 액세스 관리 672
대상 672
ID를 통한 인증 673
정책을 사용하여 액세스 관리 676
IAM AWS 서비스 작업 방식 678
AWS 자격 증명 및 액세스 문제 해결 678
규정 준수 검증 680
복원성 681
인프라 보안 681
문서 기록 683
..... dclxxxvi

란 무엇입니까 AWS SDK for Kotlin?

는 Amazon Web ServicesAPIs용 Kotlin을 AWS SDK for Kotlin 제공합니다. 를 사용하여 Amazon S3SDK, Amazon , Amazon EC2 DynamoDB 등과 함께 작동하는 Kotlin 애플리케이션을 빌드할 수 있습니다. Amazon S3 Kotlin를 사용하면 JVM 플랫폼 또는 Android API 레벨 24 이상을 대상으로 SDK지정할 수 있습니다. 및 네이티브 JavaScript 와 같은 추가 플랫폼에 대한 지원이 향후 릴리스에서 제공될 예정입니다.

향후 릴리스에서 예정된 기능을 추적하려면 의 로드맵 GitHub을 참조하세요.

SDK 시작

를 시작하려면 시작 자습서를 SDK따르세요.

개발 환경을 설정하려면 설정을 참조하세요.

요청을 하기 위한 서비스 클라이언트를 생성하고 구성하려면 구성을 AWS 서비스참조하세요. 의 다양한 기능에 대한 자세한 내용은 섹션을 SDK참조하세요SDK 사용.

사용 사례 및 특정 API 작업 수행 예제는 섹션을 참조하세요코드 예제.

SDK 메이저 버전에 대한 유지 관리 및 지원

SDK 주요 버전 및 기본 종속성에 대한 유지 관리 및 지원에 대한 자세한 내용은 AWS SDKs 및 도구 참조 안내서의 다음 주제를 참조하세요.

- AWS SDKs 및 도구 유지 관리 정책
- AWS SDKs 및 도구 버전 지원 매트릭스

추가 리소스

이 가이드 외에도 다음은 SDK Kotlin 개발자에게 유용한 온라인 리소스입니다.

- AWS 개발자 블로그
- 개발자 포럼
- SDK 소스(GitHub)

- AWS 코드 샘플 카탈로그
- @awsdevelopers(X, 이전 Twitter)

SDK for Kotlin 시작하기

는 각각에 대한 Kotlin APIs AWS SDK for Kotlin 제공합니다 AWS 서비스. SDK를 사용하면 Amazon S3, Amazon EC2, Amazon DynamoDB 등과 함께 작동하는 Kotlin 애플리케이션을 구축할 수 있습니다.

이 자습서에서는 Gradle을 사용하여에 대한 종속성을 정의하는 방법을 보여줍니다 AWS SDK for Kotlin. 그런 다음 DynamoDB 테이블에 데이터를 쓰는 코드를 생성합니다. IDE의 기능을 사용하려는 경우이 자습서에 필요한 것은 터미널 창과 텍스트 편집기뿐입니다.

자습서를 완료하려면 이 단계를 따릅니다.

- 1단계: 튜토리얼 설정
- 2단계: 프로젝트 생성
- 3단계: 코드 작성
- 4단계: 애플리케이션 빌드 및 실행

1단계: 튜토리얼 설정

이 자습서를 시작하기 전에 DynamoDB에 액세스할 수 있는 IAM Identity Center 권한 세트와에 액세스하려면 IAM Identity Center Single Sign-On 설정으로 구성된 Kotlin 개발 환경이 필요합니다 AWS.

이 가이드기본 설정의에 있는 지침에 따라이 자습서의 기본 설정을 가져옵니다.

Kotlin SDK에 대한 Single Sign-On 액세스로 개발 환경을 구성하고 활성 AWS 액세스 포털 세션이 있으면 2단계를 진행합니다.

2단계: 프로젝트 생성

이 자습서의 프로젝트를 생성하려면 먼저 Gradle을 사용하여 Kotlin 프로젝트의 기본 파일을 생성합니다. 그런 다음에 필요한 설정, 종속성 및 코드로 파일을 업데이트합니다 AWS SDK for Kotlin.

Gradle을 사용하여 새 프로젝트를 생성하려면

> **Note**
>
> 이 자습서에서는 `gradle init` 명령과 함께 Gradle 버전 8.11.1을 사용하며, 아래 3단계에서 5개의 프롬프트를 제공합니다. 다른 버전의 Gradle을 사용하는 경우 프롬프트는 사전 채워진 아티팩트 버전뿐만 아니라 및 도 다를 수 있습니다.

1. 데스크톱 또는 홈 폴더와 같이 원하는 위치에 `getstarted` 라는 새 디렉터리를 생성합니다.
2. 터미널 또는 명령 프롬프트 창을 열고 생성한 `getstarted` 디렉터리로 이동합니다.
3. 다음 명령을 사용하여 새 Gradle 프로젝트와 기본 Kotlin 클래스를 생성합니다.

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

   - 대상에 대한 메시지가 표시되면 `Enter` (기본값은)를 `Java version`누릅니다21.
   - 와 함께 메시지가 표시되면 `Enter` (이 자습서`getstarted`에서는 디렉터리 이름 기본값)을 `Project name`누릅니다.
   - 에 대한 메시지가 표시되면 `Enter` (기본값은)를 `application structure`누릅니다`Single application project`.
   - 와 함께 메시지가 표시되면 `Enter` (기본값은)를 `Select test framework`누릅니다`kotlin.test`.
   - 와 함께 메시지가 표시되면 `Enter` (기본값은)를 `Generate build using new APIs and behavior`누릅니다no.

및 Amazon S3에 AWS SDK for Kotlin 대한 종속성을 사용하여 프로젝트를 구성하려면

- 이전 절차에서 생성한 `getstarted` 디렉터리에서 `settings.gradle.kts` 파일의 콘텐츠를 다음 콘텐츠로 바꾸고 *`X.Y.Z# ## ### SDK for Kotlin`*으로 바꿉니다. https://github.com/awslabs/aws-sdk-kotlin/releases/latest

```
dependencyResolutionManagement {
    repositories {
        mavenCentral()
    }

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

```
        }
    }
}

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

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

- gradle 디렉터리 내부의 getstarted 디렉터리로 이동합니다. 라는 버전 카탈로그 파일의 내용을 다음 내용libs.versions.toml으로 바꿉니다.

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

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

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

- app 디렉터리로 이동하여 build.gradle.kts 파일을 엽니다. 그 내용을 다음 코드로 교체하고 변경 내용을 저장합니다.

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


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

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

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

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

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

dependencies 섹션에는의 Amazon S3 모듈에 대한 implementation 항목이 포함되어 있습니다 AWS SDK for Kotlin. Gradle 컴파일러는 java 섹션의 Java 21을 사용하도록 구성됩니다.

3단계: 코드 작성

프로젝트를 생성하고 구성한 후 다음 예제 코드를 App 사용하도록 프로젝트의 기본 클래스를 편집합니다.

1. 프로젝트 폴더에서 디렉터리 로 app이동합니다src/main/kotlin/org/example. App.kt 파일을 엽니다.
2. 콘텐츠를 다음 코드로 바꾸고 파일을 저장합니다.

```
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!")
```

```
}
```

4단계: 애플리케이션 빌드 및 실행

프로젝트가 생성되고 예제 클래스가 포함된 후 애플리케이션을 빌드하고 실행합니다.

1. 터미널 또는 명령 프롬프트 창을 열고 프로젝트 디렉토리 `getstarted`로 이동합니다.
2. 다음 명령을 사용하여 애플리케이션을 빌드하고 실행합니다.

```
gradle run
```

> **Note**
>
> 를 받으면 활성 Single Sign-On 세션이 없을 `IdentityProviderException`수 있습니다. `aws sso login` AWS CLI 명령을 실행하여 새 세션을 시작합니다.

애플리케이션은 createBucket API 작업을 호출하여 새 S3 버킷을 생성한 다음 putObject를 호출하여 새 객체를 새 S3 버킷에 넣습니다.

끝에 있는 `cleanUp()` 함수에서 애플리케이션은 객체를 삭제한 다음 S3 버킷을 삭제합니다.

Amazon S3 콘솔에서 결과를 보려면

1. 에서 `runBlocking` 섹션의 줄`cleanUp(s3)`에 `App.kt`주석을 추가하고 파일을 저장합니다.
2. 를 실행하여 프로젝트를 다시 빌드하고 새 객체를 새 S3 버킷에 넣습니다`gradle run`.
3. Amazon S3 콘솔에 로그인하여 새 S3 버킷에서 새 객체를 봅니다.

객체를 본 후 S3 버킷을 삭제합니다.

Success

Gradle 프로젝트가 오류 없이 빌드되고 실행되면가 축하합니다. 를 사용하여 첫 번째 Kotlin 애플리케이션을 성공적으로 빌드했습니다 AWS SDK for Kotlin.

정리

새 애플리케이션 개발을 마쳤으면 요금이 발생하지 않도록이 자습서 중에 생성한 AWS 리소스를 삭제합니다. 2단계에서 생성한 프로젝트 폴더(`get-started`)를 삭제하거나 아카이브할 수도 있습니다.

다음 단계에 따라 리소스를 정리합니다.

- `cleanUp()` 함수에 대한 호출에 주석을 추가한 경우 Amazon S3 콘솔을 사용하여 S3 버킷을 삭제합니다. Amazon S3

다음 단계

이제 기본 사항을 갖추었으므로, 다음 내용을 배울 수 있습니다.

- SDK for Kotlin을 사용하기 위한 추가 설정 단계
- Kotlin용 SDK 구성
- SDK for Kotlin 사용
- SDK for Kotlin에 대한 보안

설정 AWS SDK for Kotlin

를 AWS 서비스 사용하여를 요청하려면 다음이 AWS SDK for Kotlin필요합니다.

- AWS 액세스 포털에 로그인하는 기능
- 애플리케이션에 필요한 AWS 리소스를 사용할 수 있는 권한
- 다음 요소가 포함된 개발 환경
  - 다음 방법 중 하나 이상으로 설정된 공유 구성 파일:
    - `config` 파일에는 SDK가 자격 증명을 가져올 수 있도록 IAM Identity Center AWS 자격 증명 설정이 포함되어 있습니다.
    - `credentials` 파일에 임시 자격 증명이 포함되어 있습니다.
  - Gradle 또는 Maven과 같은 빌드 자동화 도구
- 애플리케이션을 실행할 준비가 되면 활성 AWS 액세스 포털 세션

이 주제의 내용

- 기본 설정
- 프로젝트 빌드 파일 생성
- SDK for Kotlin을 사용하여 Kotlin 프로젝트 코딩

기본 설정

개요

를 AWS 서비스 사용하여에 액세스하는 애플리케이션을 성공적으로 개발하려면 AWS SDK for Kotlin 다음 요구 사항을 충족해야 합니다.

- AWS IAM Identity Center에서 사용할 수 있는 AWS 액세스 포털에 로그인할 수 있어야 합니다.
- SDK에 대해 구성된 IAM 역할의 권한은 애플리케이션에 필요한 AWS 서비스 에 대한 액세스를 허용해야 합니다. PowerUserAccess AWS 관리형 정책과 관련된 권한은 대부분의 개발 요구 사항에 충분합니다.
- 다음 요소가 포함된 개발 환경
  - 최소 다음 중 하나와 같은 방식으로 설정된 공유 구성 파일:

    - 이 `config` 파일에는 SDK가 AWS 자격 증명을 가져올 수 있도록 IAM Identity Center 싱글 사인온 설정이 포함되어 있습니다.
    - 이 `credentials` 파일에는 임시 자격 증명이 들어 있습니다.
- Java 8 이상 버전 설치.
- Maven 또는 Gradle과 같은 빌드 자동화 도구.
- 코드 작업을 위한 텍스트 편집기.
- (선택 사항이지만 권장) IntelliJ IDEA 또는 Eclipse와 같은 통합 개발 환경(IDE)입니다.

  IDE를 사용하는 경우 AWS Toolkit를 더 쉽게 사용할 수 있도록 통합할 수도 있습니다 AWS 서비스. AWS Toolkit for IntelliJ 및는 사용할 수 있는 두 개의 도구 키트AWS Toolkit for Eclipse입니다.

- 애플리케이션을 실행할 준비가 되면 활성 AWS 액세스 포털 세션입니다. AWS Command Line Interface 를 사용하여 IAM Identity Center의 AWS 액세스 포털에 대한 로그인 프로세스를 시작합니다.

> **Important**
>
> 이 설정 섹션의 지침에서는 사용자 또는 조직이 IAM Identity Center를 사용한다고 가정합니다. 조직에서 IAM Identity Center와 독립적으로 작동하는 외부 자격 증명 공급자를 사용하는 경우 SDK for Kotlin에서 사용할 임시 자격 증명을 가져오는 방법을 알아봅니다. 다음 지침에 따라 `~/.aws/credentials` 파일에 임시 자격 증명을 추가합니다.
> ID 제공자가 임시 자격 증명을 `~/.aws/credentials` 파일에 자동으로 추가하는 경우 SDK 또는 AWS CLI에 프로필 이름을 제공할 필요가 없도록 프로필 이름을 `[default]`으로 지정해야 합니다.

AWS 액세스 포털에 로그인 기능

AWS 액세스 포털은 IAM Identity Center에 수동으로 로그인하는 웹 위치입니다. URL의 형식은 `d-xxxxxxxxxx.awsapps.com/start` 또는 입니다*`your_subdomain`*`.awsapps.com/start`.

AWS 액세스 포털에 익숙하지 않은 경우 AWS SDKs 및 도구 참조 가이드의 IAM Identity Center 인증 주제에 있는 계정 액세스 지침을 따르세요.

SDK에 Single Sign-On 액세스 설정

SDK가 IAM Identity Center 인증을 사용하도록 하려면 프로그래밍 방식 액세스 단원의 2단계를 완료한 후 시스템에 다음 요소가 포함되어야 합니다.

- 애플리케이션을 실행하기 전에 AWS 액세스 포털 세션을 시작하는 데 AWS CLI사용하는 .
- 기본 프로필이 포함된 `~/.aws/config` 파일. SDK for Kotlin은에 요청을 보내기 전에 프로필의 SSO 토큰 공급자 구성을 사용하여 자격 증명을 획득합니다 AWS. `sso_role_name` 값은 IAM 신원 센터 권한 집합에 연결된 IAM 역할로, 애플리케이션에서 사용되는 AWS 서비스 서비스에 대한 액세스를 허용해야 합니다.

  다음 샘플 `config` 파일은 SSO 토큰 공급자 구성으로 설정된 기본 프로필을 보여줍니다. 프로필의 `sso_session` 설정은 이름이 지정된 `sso-session` 섹션을 참조합니다. `sso-session` 섹션에는 AWS 액세스 포털 세션을 시작하는 설정이 포함되어 있습니다.

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

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

SSO 토큰 공급자 구성에 사용되는 설정에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 SSO 토큰 공급자 구성을 참조하세요.

개발 환경이 전에 표시된 것처럼 프로그래밍 방식으로 액세스할 수 있도록 설정되지 않은 경우 SDK 참조 가이드의 2단계를 따르세요.

를 사용하여 로그인 AWS CLI

에 액세스하는 애플리케이션을 실행하기 전에 SDK가 IAM Identity Center 인증을 사용하여 자격 증명을 확인하려면 활성 AWS 액세스 포털 세션이 AWS 서비스필요합니다. 에서 다음 명령을 실행 AWS CLI 하여 AWS 액세스 포털에 로그인합니다.

```
aws sso login
```

기본 프로필 설정이 있으므로 `--profile` 옵션으로 명령을 호출할 필요가 없습니다. SSO 토큰 공급자 구성이 명명된 프로파일을 사용하는 경우 명령은 입니다`aws sso login --profile named-profile`.

이미 활성 세션이 있는지 테스트하려면 다음 AWS CLI 명령을 실행합니다.

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

이 명령에 대한 응답은 공유 `config` 파일에 구성된 IAM Identity Center 계정 및 권한 집합을 보고해야 합니다.

> **Note**
>
> 이미 활성 AWS 액세스 포털 세션이 있고 `aws sso login`을 실행하는 경우 보안 인증 정보를 제공하지 않아도 됩니다.
> 하지만 정보에 액세스할 수 있는 `botocore` 권한을 요청하는 대화 상자가 표시됩니다.
> `botocore`는 AWS CLI 기반이 됩니다.
> 허용을 선택하여 AWS CLI 및 SDK for Kotlin에 대한 정보에 대한 액세스를 승인합니다.

Java 및 빌드 도구 설치

개발 환경에 다음이 있어야 합니다.

- JDK 8 이상. 는 Oracle Java SE 개발 키트 및 Amazon Corretto, Red Hat OpenJDK, AdoptOpenJDK와 같은 Open Java 개발 키트(OpenJDK) 배포와 함께 AWS SDK for Kotlin 작동합니다. OpenJDK
- Apache Maven, Gradle 또는 IntelliJ와 같은 Maven Central을 지원하는 빌드 도구 또는 IDE.
  - Maven 설치 및 사용 방법에 대한 자세한 내용은 http://maven.apache.org/을 참조하세요.
  - Gradle 설치 및 사용 방법에 대한 자세한 내용은 https://gradle.org/을 참조하세요.
  - IntelliJ IDEA 설치 및 사용 방법에 대한 자세한 내용은 https://www.jetbrains.com/idea/을 참조하세요.

임시 자격 증명 사용

SDK에 대한 IAM Identity Center Single Sign-On 액세스를 구성하는 대신 임시 자격 증명으로 개발 환경을 구성할 수 있습니다.

임시 자격 증명 파일을 위한 로컬 자격 증명 파일 설정

1. 공유 자격 증명 파일 생성
2. 자격 증명 파일에 임시 자격 증명을 붙여넣을 때까지 다음 자리 표시자 텍스트를 붙여 넣습니다.

```
[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. 파일을 저장합니다. 이제 파일 `~/.aws/credentials`이 로컬 개발 시스템에 존재해야 합니다. 이 파일에는 이름이 지정된 특정 프로파일이 지정되지 않은 경우 SDK for Kotlin이 사용하는 [기본값] 프로파일이 포함되어 있습니다.
4. AWS 액세스 포털에 로그인
5. 수동 자격 증명 새로 고침 제목의 다음 지침에 따라 AWS 액세스 포털에서 IAM 역할 자격 증명을 복사합니다.

   a. 링크된 지침의 4단계에서 개발 요건에 액세스 권한을 부여하는 IAM 역할 이름을 선택합니다. 이 역할은 일반적으로 PowerUserAccess 또는 Developer와 같은 이름을 갖습니다.

   b. 7단계에서 AWS 자격 증명 파일에 수동으로 프로필 추가 옵션을 선택하고 내용을 복사합니다.

6. 복사한 자격 증명을 로컬 `credentials` 파일에 붙여넣고 붙여넣은 프로필 이름을 모두 제거합니다. 파일은 다음과 유사해야 합니다.

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

7. `credentials` 파일을 저장합니다.

SDK for Kotlin은 서비스 클라이언트를 생성하고 각 요청에 사용할 때 이러한 임시 자격 증명에 액세스합니다. 5a단계에서 선택한 IAM 역할 설정에 따라 임시 보안 인증의 유효 기간이 결정됩니다. 최대 유효 기간은 12시간입니다.

임시 보안 인증이 만료되면 4~7단계를 반복합니다.

프로젝트 빌드 파일 생성

Single Sign-On 액세스 및 개발 환경을 구성한 후 원하는 빌드 도구를 사용하여 Kotlin 프로젝트를 생성합니다. 빌드 파일에서 애플리케이션이 액세스해야 AWS 서비스 하는의 종속성을 지정합니다.

가능한 모든 Maven 아티팩트 이름 목록을 보려면 API 참조 설명서를 참조하세요. SDK의 최신 버전을 찾으려면 GitHub에서 최신 릴리스를 확인하세요.

다음 샘플 빌드 파일은 7로 프로젝트 코딩을 시작하는 데 필요한 요소를 제공합니다 AWS 서비스.

Gradle

는 종속성 이름을 검색하고 버전 번호를 여러 아티팩트에서 동기화된 상태로 유지하는 데 도움이 되는 Gradle 버전 카탈로그 및 BOM(자재 명세서)을 AWS SDK for Kotlin 게시합니다.

버전 카탈로그는 버전 8 이전의 Gradle의 미리 보기 기능입니다. 사용하는 Gradle 버전에 따라 특성 미리 보기 API를 통해 옵트인해야 할 수 있습니다.

Gradle 버전 카탈로그를 사용하려면

1. `settings.gradle.kts` 파일에서 `versionCatalogs` 블록 내에 `dependencyResolutionManagement` 블록을 추가합니다.

   다음 예제 파일은 AWS SDK for Kotlin 버전 카탈로그를 구성합니다. *X.Y.Z* 링크로 이동하여 사용 가능한 최신 버전을 볼 수 있습니다.

```
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. 버전 카탈로그에서 사용할 수 있는 유형 안전 식별자`build.gradle.kts`를 사용하여에서 종속성을 선언합니다.

   다음 예제 파일은 7개의에 대한 종속성을 선언합니다 AWS 서비스.

```
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 버전, 예: 17 또는 21.

Maven

다음 예제 pom.xml 파일에는 7개의에 대한 종속성이 있습니다 AWS 서비스. *X.Y.Z* 링크로 이동하여 사용 가능한 최신 버전을 확인할 수 있습니다.

```
<?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 버전, 예: 17 또는 21.

SDK for Kotlin을 사용하여 Kotlin 프로젝트 코딩

이제 재미가 시작됩니다. 애플리케이션을 개발할 때 AWS SDK for Kotlin API 작업에 대한 전체 정보는 API 참조를 참조할 수 있습니다. 일반 Kotlin API 정보는 다음 링크를 사용합니다.

- 표준 라이브러리 API 참조
- Coroutines 개요
- Coroutines API

를 사용하여 로그인 AWS CLI

에 액세스하는 프로그램을 실행할 때마다 활성 AWS 액세스 포털 세션이 AWS 서비스필요합니다. 다음 명령을 실행하여 이 작업을 수행할 수 있습니다.

```
aws sso login
```

기본 프로필이 설정되어 있으므로 `--profile` 옵션으로 명령을 호출할 필요가 없습니다. IAM Identity Center Single Sign-On 구성에서 명명된 프로파일을 사용하는 경우 명령은 입니다`aws sso login --profile named-profile`.

이미 활성 세션이 있는지 테스트하려면 다음 AWS CLI 명령을 실행합니다.

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

이 명령에 대한 응답은 공유 `config` 파일에 구성된 IAM Identity Center 계정 및 권한 집합을 보고해야 합니다.

구성 AWS SDK for Kotlin

이 섹션에서는를 사용하여 서비스 클라이언트를 구성하는 방법을 설명합니다 AWS SDK for Kotlin. 자세한 내용은 모든SDK에 적용되는 구성 개요가 포함된 SDK 및 도구 참조 가이드를 참조하세요. AWS SDKs

목차

- 서비스 클라이언트 생성
  - 코드로 클라이언트 구성
  - 환경에서 클라이언트 구성
  - 클라이언트 닫기
- AWS 리전 선택
  - 기본 리전 공급자 체인
- 보안 인증 제공업체
  - 기본 자격 증명 공급자 체인
    - 기본 자격 증명 공급자 체인에 대해 알아보기
  - 명시적 자격 증명 공급자
- 클라이언트 엔드포인트 구성
  - 사용자 지정 구성
    - endpointUrl 설정
    - endpointProvider 설정
      - EndpointProvider 속성
    - endpointUrl 또는 endpointProvider
    - Amazon S3에 대한 참고 사항
  - 예시
    - endpointUrl 예제
    - endpointProvider 예제
    - endpointUrl 및 endpointProvider
- HTTP
  - HTTP 클라이언트 구성
    - 기본 구성

        - 가져오기
        - 코드
    - HTTP 엔진 유형 지정
        - 가져오기
        - 코드
        - OkHttp4Engine 사용
        - 명시적 HTTP 클라이언트 사용
            - 가져오기
            - 코드
- HTTP 프록시 사용
    - JVM 시스템 속성 사용
    - 환경 변수 사용
    - EC2 인스턴스에서 프록시 사용
- HTTP 인터셉터
    - 인터셉터 등록
        - 모든 서비스 클라이언트 작업에 대한 인터셉터
        - 특정 작업 전용 인터셉터
- 최소 TLS 버전 적용
    - HTTP 엔진 구성
    - sdk.minTls JVM 시스템 속성 설정
    - SDK_MIN_TLS 환경 변수 설정
- 재시도
    - 기본 재시도 구성
    - 최대 시도 횟수
    - 지연 및 백오프
    - 토큰 버킷 재시도
    - 적응형 재시도
- 관찰성
- 구성 TelemetryProvider
    - 기본 글로벌 원격 측정 공급자 구성

        - 특정 서비스 클라이언트에 대한 원격 측정 공급자 구성
    - Metrics
    - 로깅
        - 와이어 수준 메시지에 대한 로그 모드 지정
            - 코드에서 로그 모드 설정
            - 환경에서 로그 모드 설정
    - 원격 측정 공급자
        - OpenTelemetry 기반 원격 측정 공급자 구성
            - 사전 조건
            - SDK 구성
            - 리소스
- 서비스 클라이언트 구성 재정의
    - 재정의된 클라이언트의 수명 주기
    - 클라이언트 간에 공유되는 리소스

서비스 클라이언트 생성

에 요청하려면 먼저 해당 서비스의 클라이언트를 인스턴스화 AWS 서비스해야 합니다.

HTTP 클라이언트가 사용, 로깅 수준 및 재시도 구성과 같은 서비스 클라이언트에 대한 공통 설정을 구성할 수 있습니다. 또한 각 서비스 클라이언트에는 AWS 리전 및 자격 증명 공급자가 필요합니다. SDK는 이러한 값을 사용하여 요청을 올바른 리전으로 보내고 올바른 자격 증명으로 요청에 서명합니다.

이러한 값을 프로그래밍 방식으로 코드로 지정하거나 환경에서 자동으로 로드할 수 있습니다.

코드로 클라이언트 구성

특정 값으로 서비스 클라이언트를 구성하려면 다음 코드 조각과 같이 서비스 클라이언트 팩토리 메서드에 전달되는 lambda 함수에서 해당 값을 지정할 수 있습니다.

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

구성 블록에서 지정하지 않은 모든 값은 기본값으로 설정됩니다. 예를 들어 이전 코드와 같이 자격 증명 공급자를 지정하지 않으면 자격 증명 공급자가 기본 자격 증명 공급자 체인으로 기본 설정됩니다.

> **Warning**
>
> 와 같은 일부 속성에는 기본값이 `region` 없습니다. 프로그래밍 방식 구성을 사용할 때는 구성 블록에서 명시적으로 지정해야 합니다. SDK가 속성을 확인할 수 없는 경우 API 요청이 실패할 수 있습니다.

환경에서 클라이언트 구성

서비스 클라이언트를 생성할 때 SDK는 현재 실행 환경의 위치를 검사하여 일부 구성 속성을 확인할 수 있습니다. 이러한 위치에는 공유 구성 및 자격 증명 파일, 환경 변수 및 JVM 시스템 속성이 포함됩니다. 확인할 수 있는 속성에는 AWS 리전, 재시도 전략, 로그 모드 등이 포함됩니다. SDK가 실행 환경에서 확인할 수 있는 모든 설정에 대한 자세한 내용은 AWS SDKs 및 도구 설정 참조 가이드를 참조하세요.

환경 기반 구성으로 클라이언트를 생성하려면 서비스 클라이언트 인터페이스`suspend fun fromEnvironment()`에서 정적 메서드를 사용합니다.

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

이러한 방식으로 클라이언트를 생성하면 Amazon EC2 AWS Lambda또는 환경에서 서비스 클라이언트의 구성을 사용할 수 있는 다른 컨텍스트에서 실행할 때 유용합니다. 이렇게 하면 코드를 실행 중인 환경에서 분리하여 코드를 변경하지 않고도 애플리케이션을 여러 리전에 더 쉽게 배포할 수 있습니다.

또한 Lambda 블록을에 전달하여 특정 속성을 재정의할 수 있습니다`fromEnvironment`. 다음 예제에서는 환경(예: 리전)에서 일부 구성 속성을 로드하지만 특히 프로필의 자격 증명을 사용하도록 자격 증명 공급자를 재정의합니다.

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

SDK는 프로그래밍 방식 설정 또는 환경에서 확인할 수 없는 구성 속성에 기본값을 사용합니다. 예를 들어 코드 또는 환경 설정에서 자격 증명 공급자를 지정하지 않으면 자격 증명 공급자가 기본 자격 증명 공급자 체인으로 기본 설정됩니다.

> **Warning**
>
> 리전과 같은 일부 속성에는 기본값이 없습니다. 환경 설정에서 지정하거나 구성 블록에서 명시적으로 지정해야 합니다. SDK가 속성을 확인할 수 없는 경우 API 요청이 실패할 수 있습니다.

> **Note**
>
> 임시 액세스 키 및 SSO 구성과 같은 자격 증명 관련 속성은 실행 환경에서 찾을 수 있지만 생성 시 클라이언트가 값을 소싱하지 않습니다. 대신 각 요청 시 자격 증명 공급자 계층에서 값에 액세스합니다.

클라이언트 닫기

서비스 클라이언트가 더 이상 필요하지 않은 경우 이를 닫아 사용 중인 리소스를 해제합니다.

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

서비스 클라이언트는 `Closeable` 인터페이스를 확장하므로 다음 코드 조각과 같이 `use` 확장을 사용하여 블록 끝에 클라이언트를 자동으로 닫을 수 있습니다.

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

이전 예제에서 lambda 블록은 방금 생성된 클라이언트에 대한 참조를 수신합니다. 이 클라이언트 참조에 대한 작업을 호출하고 예외 발생을 포함하여 블록이 완료되면 클라이언트가 닫힐 수 있습니다.

AWS 리전 선택

를 사용하면 특정 지리적 영역에서 AWS 서비스 작동하는에 액세스할 AWS 리전수 있습니다. 이는 중복성은 물론, 데이터와 애플리케이션을 고객과 고객의 사용자가 액세스할 위치에 가까운 곳에서 실행 상태로 유지하는 데도 유용할 수 있습니다.

기본 리전 공급자 체인

환경에서 서비스 클라이언트의 구성을 로드할 때 다음 조회 프로세스가 사용됩니다.

1. 빌더에 설정된 명시적 리전입니다.
2. `aws.region` JVM 시스템 속성이 확인됩니다. 설정된 경우 해당 리전은 클라이언트 구성에 사용됩니다.
3. `AWS_REGION` 환경 변수를 확인합니다. 설정된 경우 해당 리전은 클라이언트 구성에 사용됩니다.
   a. 참고:이 환경 변수는 Lambda 컨테이너에 의해 설정됩니다.
4. SDK는 AWS 공유 구성 파일을 확인합니다. 활성 프로필에 속성`region`이 설정된 경우 SDK가 속성을 사용합니다.
   a. `AWS_CONFIG_FILE` 환경 변수는 공유 구성 파일의 위치를 사용자 지정하는 데 사용할 수 있습니다.
   b. `aws.profile` JVM 시스템 속성 또는 `AWS_PROFILE` 환경 변수를 사용하여 SDK가 로드하는 프로파일을 사용자 지정할 수 있습니다.
5. SDK는 Amazon EC2 인스턴스 메타데이터 서비스를 사용하여 현재 실행 중인 EC2 인스턴스의 리전을 확인하려고 시도합니다.
6. 이 시점에서 리전이 여전히 해결되지 않으면 예외를 제외하고 클라이언트 생성이 실패합니다.

보안 인증 제공업체

> ⚠ 기본 자격 증명 공급자 체인이 버전 1.4.0으로 변경된 자격 증명을 확인하는 순서 자세한 내용은 아래 참고 사항을 참조하세요.

를 사용하여 Amazon Web Services에 요청하기 위해 SDK AWS SDK for Kotlin는에서 발급한 암호화 서명 보안 인증을 사용합니다 AWS. SDK는 자격 증명을 획득하기 위해 JVM 시스템 속성, 환경 변수, 공유 AWS `config` 및 `credentials` 파일, Amazon EC2 인스턴스 메타데이터 등 여러 위치에 있는 구성 설정을 사용할 수 있습니다.

SDK는 자격 증명 공급자 추상화를 사용하여 다양한 소스에서 자격 증명을 검색하는 프로세스를 간소화합니다. SDK에는 여러 자격 증명 공급자 구현이 포함되어 있습니다.

예를 들어 검색된 구성에 공유 `config` 파일의 IAM Identity Center Single Sign-On 액세스 설정이 포함된 경우 SDK는 IAM Identity Center와 협력하여 요청에 사용하는 임시 자격 증명을 검색합니다 AWS

서비스. 이러한 자격 증명 획득 접근 방식을 통해 SDK는 IAM Identity Center 공급자(SSO 자격 증명 공급자라고도 함)를 사용합니다. 이 가이드의 설정 섹션에서는이 구성을 설명합니다.

특정 자격 증명 공급자를 사용하려면 서비스 클라이언트를 생성할 때 자격 증명 공급자를 지정할 수 있습니다. 또는 기본 자격 증명 공급자 체인을 사용하여 구성 설정을 자동으로 검색할 수 있습니다.

기본 자격 증명 공급자 체인

클라이언트 구성에서 명시적으로 지정되지 않은 경우 SDK for Kotlin은 자격 증명을 제공할 수 있는 각 위치를 순차적으로 확인하는 자격 증명 공급자를 사용합니다. 이 기본 자격 증명 공급자는 자격 증명 공급자 체인으로 구현됩니다.

기본 체인을 사용하여 애플리케이션에서 자격 증명을 제공하려면 `credentialsProvider` 속성을 명시적으로 제공하지 않고 서비스 클라이언트를 생성합니다.

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

서비스 클라이언트 생성에 대한 자세한 내용은 클라이언트 구성 및 구성을 참조하세요.

기본 자격 증명 공급자 체인에 대해 알아보기

기본 자격 증명 공급자 체인은 다음과 같은 사전 정의된 시퀀스를 사용하여 자격 증명 구성을 검색합니다. 구성된 설정이 유효한 자격 증명을 제공하면 체인이 중지됩니다.

1. AWS 액세스 키(JVM 시스템 속성)

   SDK는 `aws.accessKeyId`, `aws.secretAccessKey`및 `aws.sessionToken` JVM 시스템 속성을 찾습니다.

2. AWS 액세스 키(환경 변수)

   SDK는 `AWS_ACCESS_KEY_ID` 및 환경 `AWS_SECRET_ACCESS_KEY` `AWS_SESSION_TOKEN` 변수에서 자격 증명을 로드하려고 시도합니다.

3. 웹 자격 증명 토큰

   SDK는 환경 변수 `AWS_WEB_IDENTITY_TOKEN_FILE` 및 `AWS_ROLE_ARN` (또는 JVM 시스템 속성 `aws.webIdentityTokenFile` 및 `aws.roleArn`)를 찾습니다. 토큰 정보와 역할에 따라 SDK는 임시 자격 증명을 획득합니다.

4. 구성 파일의 프로필

   이 단계에서 SDK는 프로필과 연결된 설정을 사용합니다. 기본적으로 SDK는 공유 AWS `config` 및 `credentials` 파일을 사용하지만 `AWS_CONFIG_FILE` 환경 변수가 설정된 경우 SDK는 해당 값을 사용합니다. `AWS_PROFILE` 환경 변수(또는 `aws.profile` JVM 시스템 속성)가 설정되지 않은 경우 SDK는 "기본" 프로파일을 찾고, 그렇지 않으면 `AWS_PROFILE's` 값과 일치하는 프로파일을 찾습니다.

   SDK는 이전 단락에 설명된 구성을 기반으로 프로파일을 찾고 여기에 정의된 설정을 사용합니다. SDK에서 찾은 설정에 서로 다른 자격 증명 공급자 접근 방식에 대한 설정 조합이 포함된 경우 SDK는 다음 순서를 사용합니다.

   a. AWS 액세스 키(구성 파일) - SDK는 `aws_access_key_id`, `aws_access_key_id`및에 대한 설정을 사용합니다`aws_session_token`.

   b. 역할 구성 수임 - SDK가 `role_arn` 및 `source_profile` 또는 `credential_source` 설정을 찾으면 역할을 수임하려고 시도합니다. SDK가 `source_profile` 설정을 찾으면 다른 프로필에서 자격 증명을 소싱하여에서 지정한 역할에 대한 임시 자격 증명을 수신합니다`role_arn`. SDK가 `credential_source` 설정을 찾으면 `credential_source` 설정 값에 따라 Amazon ECS 컨테이너, Amazon EC2 인스턴스 또는 환경 변수에서 자격 증명을 가져옵니다. 그런 다음 해당 자격 증명을 사용하여 역할에 대한 임시 자격 증명을 획득합니다.

      프로필에는 `source_profile` 설정 또는 `credential_source` 설정이 포함되어야 하지만 둘 다 포함될 수는 없습니다.

   c. 웹 자격 증명 토큰 구성 - SDK가 `role_arn` 및 `web_identity_token_file` 설정을 찾으면 `role_arn` 및 토큰을 기반으로 AWS 리소스에 액세스할 수 있는 임시 자격 증명을 획득합니다.

   d. SSO 토큰 구성 - SDK가 `sso_session`, `sso_account_id`, `sso_role_name` 설정(구성 파일의 컴패니언 `sso-session` 섹션과 함께)을 찾으면 SDK는 IAM Identity Center 서비스에서 임시 자격 증명을 검색합니다.

   e. 레거시 SSO 구성 - SDK가 `sso_start_url`, `sso_regionsso_account_id`, 및 `sso_role_name` 설정을 찾으면 SDK는 IAM Identity Center 서비스에서 임시 자격 증명을 검색합니다.

   f. 프로세스 구성 - SDK가 `credential_process` 설정을 찾으면 경로 값을 사용하여 프로세스를 호출하고 임시 자격 증명을 획득합니다.

5. 컨테이너 자격 증명

   SDK는 환경 변수 `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` 또는 `AWS_CONTAINER_CREDENTIALS_FULL_URI` 및

AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE 또는를 찾습니다AWS_CONTAINER_AUTHORIZATION_TOKEN. 이 값을 사용하여 GET 요청을 통해 지정된 HTTP 엔드포인트에서 자격 증명을 로드합니다.

6. IMDS 자격 증명

   SDK는 기본 또는 구성된 HTTP 엔드포인트에서 인스턴스 메타데이터 서비스에서 자격 증명을 가져오려고 시도합니다. SDK는 IMDSv2만 지원합니다.

현재 자격 증명이 여전히 해결되지 않으면 예외를 제외하고 클라이언트 생성이 실패합니다.

> **참고: 자격 증명 확인 순서 변경**
>
> 위에서 설명한 자격 증명 확인 순서는 SDK for Kotlin 1.4.x+ 릴리스의 최신 상태입니다. 1.4.0 릴리스 전에 항목 번호 3과 4가 전환되었고 현재 4a 항목은 현재 4f 항목을 따랐습니다.

명시적 자격 증명 공급자

기본 공급자 체인을 사용하는 대신 SDK에서 사용해야 하는 특정 자격 증명 공급자 또는 사용자 지정 체인(CredentialsProviderChain)을 지정할 수 있습니다. 예를 들어 환경 변수를 사용하여 기본 자격 증명을 설정하는 경우 다음 코드 조각과 같이 클라이언트 빌더EnvironmentCredentialsProvider에를 제공합니다.

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

> **Note**
>
> 기본 체인은 자격 증명을 캐시하지만 독립 실행형 공급자는 캐시하지 않습니다. CachedCredentialsProvider 클래스를 사용하여 모든 자격 증명 공급자를 래핑하여 모든 API 호출에서 자격 증명을 불필요하게 가져오지 않도록 할 수 있습니다. 캐시된 공급자는 현재 자격 증명이 만료될 때만 새 자격 증명을 가져옵니다.

> **Note**
>
> `CredentialsProvider` 인터페이스를 구현하여 자체 자격 증명 공급자 또는 공급자 체인을 구현할 수 있습니다.

클라이언트 엔드포인트 구성

가를 AWS SDK for Kotlin 호출할 때 첫 번째 단계 AWS 서비스중 하나는 요청을 라우팅할 위치를 결정하는 것입니다. 이 프로세스를 엔드포인트 해상도라고 합니다.

서비스 클라이언트를 빌드할 때 SDK에 대한 엔드포인트 확인을 구성할 수 있습니다. 엔드포인트 확인의 기본 구성은 일반적으로 좋지만 기본 구성을 수정하게 되는 몇 가지 이유가 있습니다. 두 가지 이유는 다음과 같습니다.

- 서비스의 사전 릴리스 버전 또는 서비스의 로컬 배포를 요청합니다.
- SDK에서 아직 모델링되지 않은 특정 서비스 기능에 대한 액세스.

> **Warning**
>
> 엔드포인트 확인은 고급 SDK 주제입니다. 기본 설정을 변경하면 코드가 해제될 위험이 있습니다. 기본 설정은 프로덕션 환경의 대부분의 사용자에게 적용되어야 합니다.

사용자 지정 구성

클라이언트를 빌드할 때 사용할 수 있는 두 가지 속성을 사용하여 서비스 클라이언트의 엔드포인트 확인을 사용자 지정할 수 있습니다.

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

`endpointUrl` 설정

서비스의 "기본" 호스트 이름을 나타내`endpointUrl`도록 값을 설정할 수 있습니다. 그러나이 값은 클라이언트 `EndpointProvider` 인스턴스에 파라미터로 전달되므로 최종적인 것은 아닙니다. 그런 다

음 EndpointProvider 구현은 해당 값을 검사하고 잠재적으로 수정하여 최종 엔드포인트를 결정할 수 있습니다.

예를 들어 Amazon Simple Storage Service(Amazon S3) 클라이언트에 대한 endpointUrl 값을 지정하고 GetObject 작업을 수행하는 경우 기본 엔드포인트 공급자 구현은 버킷 이름을 호스트 이름 값에 주입합니다.

실제로 사용자는 서비스의 개발 또는 미리 보기 인스턴스를 가리키도록 endpointUrl 값을 설정합니다.

endpointProvider 설정

서비스 클라이언트의 EndpointProvider 구현에 따라 최종 엔드포인트 확인이 결정됩니다. 다음 코드 블록에 표시된 EndpointProvider 인터페이스는 resolveEndpoint 메서드를 노출합니다.

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

서비스 클라이언트는 모든 요청에 대해 resolveEndpoint 메서드를 호출합니다. 서비스 클라이언트는 추가 변경 없이 공급자가 반환한 Endpoint 값을 사용합니다.

EndpointProvider 속성

resolveEndpoint 메서드는 엔드포인트 확인에 사용되는 속성이 포함된 서비스별 EndpointParameters 객체를 허용합니다.

모든 서비스에는 다음과 같은 기본 속성이 포함됩니다.

명칭	유형	설명
region	String	클라이언트의 AWS 리전
endpoint	String	값 집합의 문자열 표현 endpointUrl
useFips	불	클라이언트 구성에서 FIPS 엔드포인트가 활성화되어 있는지 여부

명칭	유형	설명
useDualStack	불	클라이언트 구성에서 듀얼 스택 엔드포인트가 활성화되었는지 여부

서비스는 해결에 필요한 추가 속성을 지정할 수 있습니다. 예를 들어 Amazon S3에는 버킷 이름과 여러 Amazon S3-specific 기능 설정이 S3EndpointParameters 포함됩니다. 예를 들어 forcePathStyle 속성에 따라 가상 호스트 주소 지정을 사용할 수 있는지 여부가 결정됩니다.

자체 공급자를 구현하는 경우 자체 인스턴스를 구성할 필요가 없습니다EndpointParameters. SDK는 각 요청에 대한 속성을 제공하고 이를 구현에 전달합니다resolveEndpoint.

endpointUrl 또는 endpointProvider

다음 두 명령문은 동등한 엔드포인트 해결 동작을 가진 클라이언트를 생성하지 않는다는 점을 이해하는 것이 중요합니다.

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

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

endpointUrl 속성을 설정하는 문은 (기본값) 공급자에 전달되는 기본 URL을 지정하며, 이는 엔드포인트 확인의 일부로 수정할 수 있습니다.

를 설정하는 문은가 S3Client 사용하는 최종 URL을 endpointProvider 지정합니다.

두 속성을 모두 설정할 수 있지만 대부분의 경우 사용자 지정이 필요한 속성 중 하나를 제공합니다. 일반 SDK 사용자는 가장 자주 endpointUrl 값을 제공합니다.

Amazon S3에 대한 참고 사항

Amazon S3는 버킷 가상 호스팅과 같은 사용자 지정 엔드포인트 사용자 지정을 통해 모델링된 많은 기능을 갖춘 복잡한 서비스입니다. 가상 호스팅은 버킷 이름이 호스트 이름에 삽입되는 Amazon S3의 기능입니다.

따라서 Amazon S3 서비스 클라이언트에서 EndpointProvider 구현을 대체하지 않는 것이 좋습니다. 해결 동작을 확장해야 하는 경우 추가 엔드포인트 고려 사항과 함께 로컬 개발 스택에 요청을 전송하여 기본 구현을 래핑하는 것이 좋습니다. 다음 endpointProvider 예제에서는이 접근 방식의 샘플 구현을 보여줍니다.

예시

endpointUrl 예제

다음 코드 조각은 Amazon S3 클라이언트에 대해 일반 서비스 엔드포인트를 재정의하는 방법을 보여줍니다.

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

endpointProvider 예제

다음 코드 조각은 Amazon S3의 기본 구현을 래핑하는 사용자 지정 엔드포인트 공급자를 제공하는 방법을 보여줍니다.

```
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` 및 `endpointProvider`

다음 예제 프로그램은 `endpointUrl`와 `endpointProvider` 설정 간의 상호 작용을 보여줍니다. 이는 고급 사용 사례입니다.

```
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

이 섹션에서는에서 HTTP 관련 설정의 구성을 다룹니다 AWS SDK for Kotlin.

주제

- HTTP 클라이언트 구성
- HTTP 프록시 사용

- HTTP 인터셉터
- 최소 TLS 버전 적용

HTTP 클라이언트 구성

기본적으로는 OkHttp 기반 HTTP 클라이언트를 AWS SDK for Kotlin 사용합니다. 명시적으로 구성된 클라이언트를 제공하여 HTTP 클라이언트와 해당 구성을 재정의할 수 있습니다.

> **Note**
>
> 기본적으로 각 서비스 클라이언트는 HTTP 클라이언트의 자체 복사본을 사용합니다. 애플리케이션에서 여러 서비스를 사용하는 경우 단일 HTTP 클라이언트를 구성하고 모든 서비스 클라이언트에서 공유할 수 있습니다.

기본 구성

서비스 클라이언트를 구성할 때 기본 엔진 유형을 구성할 수 있습니다. SDK는 결과 HTTP 클라이언트 엔진을 관리하고 더 이상 필요하지 않을 때 자동으로 닫습니다.

다음 예제는 DynamoDB 클라이언트를 초기화하는 동안 HTTP 클라이언트의 구성을 보여줍니다.

가져오기

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

코드

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

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

```
}
```

HTTP 엔진 유형 지정

고급 사용 사례의 경우 엔진 유형을 `httpClient` 지정하는 추가 파라미터를에 전달할 수 있습니다. 이렇게 하면 해당 엔진 유형에 고유한 구성 파라미터를 설정할 수 있습니다.

다음 예제에서는 `maxConcurrencyPerHost` 속성을 구성하는 데 사용할 수 `OkHttpEngine` 있는를 지정합니다.

가져오기

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

코드

```
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.
}
```

엔진 유형의 가능한 값은 `OkHttpEngine`, 및 `OkHttp4Engine`입니다`CrtHttpEngine`.

HTTP 엔진과 관련된 구성 파라미터를 사용하려면 엔진을 컴파일 시간 종속성으로 추가해야 합니다. 의 `OkHttpEngine`경우 Gradle을 사용하여 다음 종속성을 추가합니다.

(*`X.Y.Z`* 링크로 이동하여 사용 가능한 최신 버전을 볼 수 있습니다.)

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

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

에 다음 종속성을 CrtHttpEngine추가합니다.

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

OkHttp4Engine 사용

기본를 사용할 수 없는 OkHttp4Engine 경우를 사용합니다OkHttpEngine. smithy-kotlin GitHub 리포지토리에는를 구성하고 사용하는 방법에 대한 정보가 있습니다OkHttp4Engine.

명시적 HTTP 클라이언트 사용

명시적 HTTP 클라이언트를 사용하는 경우 더 이상 필요하지 않은 종료를 포함하여 해당 클라이언트의 수명에 대한 책임은 사용자에게 있습니다. HTTP 클라이언트는 최소한 이를 사용하는 서비스 클라이언트만큼 오래 지속되어야 합니다.

다음 코드 예제는 DynamoDbClient가 활성 상태인 동안 HTTP 클라이언트를 계속 유지하는 코드를 보여줍니다. use 함수는 HTTP 클라이언트가 제대로 닫히는지 확인합니다.

가져오기

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

코드

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

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

```
    }
}
```

HTTP 프록시 사용

를 사용하여 프록시 서버를 AWS 통해에 액세스하려면 JVM 시스템 속성 또는 환경 변수를 구성할 AWS SDK for Kotlin수 있습니다. 둘 다 제공되면 JVM 시스템 속성이 우선합니다.

JVM 시스템 속성 사용

SDK는 JVM 시스템 속성 `https.proxyHost`, `https.proxyPort`및를 찾습니다`http.nonProxyHosts`. 이러한 일반적인 JVM 시스템 속성에 대한 자세한 내용은 Java 설명서의 네트워킹 및 프록시를 참조하세요.

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

환경 변수 사용

SDK는 `https_proxy`, `http_proxy`및 `no_proxy` 환경 변수(및 각의 대문자 버전)를 찾습니다.

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

EC2 인스턴스에서 프록시 사용

연결된 IAM 역할로 시작된 EC2 인스턴스에서 프록시를 구성하는 경우 인스턴스 메타데이터에 액세스하는 데 사용되는 주소를 제외해야 합니다. 이렇게 하려면 `http.nonProxyHosts` JVM 시스템 속성 또는 `no_proxy` 환경 변수를 인스턴스 메타데이터 서비스의 IP 주소인 로 설정합니다`169.254.169.254`. 이 주소는 달라지지 않습니다.

```
export no_proxy=169.254.169.254
```

HTTP 인터셉터

인터셉터를 사용하여 API 요청 및 응답 실행에 연결할 수 있습니다. 인터셉터는 SDK가 요청/응답 수명 주기에 동작을 주입하기 위해 작성하는 코드를 호출하는 개방형 메커니즘입니다. 이렇게 하면 진행 중인 요청을 수정하고, 요청 처리를 디버그하고, 예외를 보는 등의 작업을 수행할 수 있습니다.

다음 예제는 재시도 루프가 입력되기 전에 모든 발신 요청에 헤더를 추가하는 간단한 인터셉터를 보여줍니다.

```
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()
    }
}
```

자세한 내용과 사용 가능한 인터셉션 후크는 인터셉터 인터페이스를 참조하세요.

인터셉터 등록

서비스 클라이언트를 구성하거나 특정 작업 세트에 대한 구성을 재정의할 때 인터셉터를 등록합니다.

모든 서비스 클라이언트 작업에 대한 인터셉터

다음 코드는 빌더의 인터셉터 속성에 AddHeader 인스턴스를 추가합니다. 이렇게 추가하면 재시도 루프가 입력되기 전에 모든 작업에 x-foo-version 헤더가 추가됩니다.

```
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 { ... }
```

특정 작업 전용 인터셉터

withConfig 확장을 사용하면 모든 서비스 클라이언트에 대한 하나 이상의 작업에 대해 서비스 클라이언트 구성을 재정의할 수 있습니다. 이 기능을 사용하면 작업 하위 집합에 대한 추가 인터셉터를 등록할 수 있습니다.

다음 예제에서는 use 확장 프로그램 내의 작업에 대한 s3 인스턴스 구성을 재정의합니다. 에서 호출된 작업s3Scoped에는 x-foo-version 및 x-bar-version 헤더가 모두 포함됩니다.

```
// '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.
}
```

최소 TLS 버전 적용

를 사용하면 서비스 엔드포인트에 연결할 때 최소 TLS 버전을 구성할 AWS SDK for Kotlin수 있습니다. SDK는 다양한 구성 옵션을 제공합니다. 가장 높은 우선순위부터 가장 낮은 우선순위까지 옵션은 다음과 같습니다.

- HTTP 엔진을 명시적으로 구성
- `sdk.minTls` JVM 시스템 속성 설정
- `SDK_MIN_TLS` 환경 변수 설정

HTTP 엔진 구성

서비스 클라이언트에 기본이 아닌 HTTP 엔진을 지정할 때 `tlsContext.minVersion` 필드를 설정할 수 있습니다.

다음 예제에서는 HTTP 엔진과 이를 사용하여 TLS v1.2를 최소한으로 사용하는 모든 서비스 클라이언트를 구성합니다.

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

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

`sdk.minTls` JVM 시스템 속성 설정

`sdk.minTls` JVM 시스템 속성을 설정할 수 있습니다. 시스템 속성 세트로 애플리케이션을 시작할 때에서 구성된 모든 HTTP 엔진은 AWS SDK for Kotlin 기본적으로 지정된 최소 TLS 버전을 사용합니다. 그러나 HTTP 엔진 구성에서 이를 명시적으로 재정의할 수 있습니다. 허용되는 값은 다음과 같습니다.

- TLS_1_0
- TLS_1_1
- TLS_1_2
- TLS_1_3

`SDK_MIN_TLS` 환경 변수 설정

`SDK_MIN_TLS` 환경 변수를 설정할 수 있습니다. 환경 변수 세트로 애플리케이션을 시작할 때 다른 옵션으로 재정의되지 않는 한에서 구성된 모든 HTTP 엔진은 지정된 최소 TLS 버전을 AWS SDK for Kotlin 사용합니다.

허용되는 값은 다음과 같습니다.

- TLS_1_0
- TLS_1_1
- TLS_1_2
- TLS_1_3

재시도

AWS 서비스 예기치 않은 예외를 가끔 반환하도록를 호출합니다. 통화를 재시도하면 제한 또는 일시적 오류와 같은 특정 유형의 오류가 성공할 수 있습니다.

이 페이지에서는를 사용하여 자동 재시도를 구성하는 방법을 설명합니다 AWS SDK for Kotlin.

기본 재시도 구성

기본적으로 모든 서비스 클라이언트는 표준 재시도 전략으로 자동으로 구성됩니다. 기본 구성은 최대 3회 실패하는 호출을 시도합니다(초기 시도 + 2회 재시도). 각 호출 간의 개입 지연은 재시도 폭풍을 방지하기 위해 지수 백오프 및 무작위 지터로 구성됩니다. 이 구성은 대부분의 사용 사례에서 작동하지만 처리량이 많은 시스템과 같은 일부 상황에서는 적합하지 않을 수 있습니다.

SDK는 재시도 가능한 오류에 대해서만 재시도를 시도합니다. 재시도 가능한 오류의 예로는 소켓 제한 시간, 서비스 측 제한, 동시성 또는 낙관적 잠금 실패, 일시적인 서비스 오류가 있습니다. 누락되거나 유효하지 않은 파라미터, 인증/보안 오류 및 잘못된 구성 예외는 재시도 가능한 것으로 간주되지 않습니다.

최대 시도 횟수, 지연 및 백오프, 토큰 버킷 구성을 설정하여 표준 재시도 전략을 사용자 지정할 수 있습니다.

최대 시도 횟수

클라이언트 구성 중에 `retryStrategy DSL 블록`에서 기본 최대 시도 횟수(3)를 사용자 지정할 수 있습니다.

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

이전 코드 조각에 표시된 DynamoDB 서비스 클라이언트를 사용하면 SDK는 최대 5회(초기 시도 + 4회 재시도) 실패하는 API 호출을 시도합니다.

다음 코드 조각과 같이 최대 시도 횟수를 1로 설정하여 자동 재시도를 완전히 비활성화할 수 있습니다.

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

지연 및 백오프

재시도가 필요한 경우 기본 재시도 전략은 후속 시도를 수행하기 전에 대기합니다. 첫 번째 재시도의 지연 시간은 작지만 이후 재시도에서는 기하급수적으로 증가합니다. 최대 지연 시간은 너무 크게 증가하지 않도록 제한됩니다.

마지막으로 모든 시도 사이의 지연에 무작위 지터가 적용됩니다. 지터는 재시도 폭풍을 일으킬 수 있는 대규모 플릿의 영향을 완화하는 데 도움이 됩니다. (지수 백오프 및 지터에 대한 자세한 내용은이 AWS 아키텍처 블로그 게시물을 참조하세요.)

지연 파라미터는 delayProvider DSL 블록에서 구성할 수 있습니다.

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

이전 코드 조각에 표시된 구성을 사용하면 클라이언트가 최대 100밀리초 동안 첫 번째 재시도를 지연합니다. 재시도 사이의 최대 시간은 5초입니다.

다음 파라미터를 사용하여 지연 및 백오프를 조정할 수 있습니다.

파라미터	기본값	설명
`initialDelay`	10밀리초	첫 번째 재시도의 최대 지연 시간입니다. 지터가 적용되면 실제 지연 시간이 더 적을 수 있습니다.
`jitter`	1.0(전체 지터)	계산된 지연을 무작위로 줄이는 최대 진폭입니다. 기본값인 1.0은 계산된 지연을 최대 100%(예: 0까지)까지 줄일 수 있음을 의미합니다. 값이 0.5이면 계산된 지연 시간을 최대 절반까지 줄일 수 있습니다. 따라서 최대 지연 시간은 10ms로 5ms에서 10ms 사이로 줄일 수 있습니다. 값이 0.0이면 지터가 적용되지 않음을 의미합니다. ⚠ Important "Jitter 구성은 고급 기능입니다. 이 동작을 사

파라미터	기본값	설명
		용자 지정하는 것은 일반적으로 권장되지 않습니다.
`maxBackoff`	20초	모든 시도에 적용할 최대 지연 시간입니다. 이 값을 설정하면 후속 시도 사이에 발생하는 지수 증가가 제한되고 계산된 최대값이 너무 커지지 않습니다. 이 파라미터는 지터가 적용되기 전에 계산된 지연을 제한합니다. 지터를 적용하면 지연 시간이 훨씬 더 줄어들 수 있습니다.
`scaleFactor`	1.5	후속 최대 지연이 증가하는 지수 기반입니다. 예를 들어가 `initialDelay` 10ms이고가 `scaleFactor` 1.5인 경우 다음과 같은 최대 지연 시간이 계산됩니다. • 재시도 1: 10ms × 1.50 = 10ms • 재시도 2: 10ms × 1.51 = 15ms • 재시도 3: 10ms × 1.52 = 22.5ms • 재시도 4: 10ms × 1.53 = 33.75ms 지터를 적용하면 각 지연의 실제 양이 더 적을 수 있습니다.

토큰 버킷 재시도

기본 토큰 버킷 구성을 조정하여 표준 재시도 전략의 동작을 추가로 수정할 수 있습니다. 재시도 토큰 버킷은 성공 가능성이 낮거나 제한 시간 초과 및 제한 실패와 같이 해결하는 데 더 많은 시간이 걸릴 수 있는 재시도를 줄이는 데 도움이 됩니다.

> **Important**
>
> 토큰 버킷 구성은 고급 기능입니다. 이 동작을 사용자 지정하는 것은 일반적으로 권장되지 않습니다.

각 재시도(선택 사항으로 초기 시도 포함)는 토큰 버킷에서 일부 용량을 줄입니다. 감소되는 양은 시도 유형에 따라 다릅니다. 예를 들어 일시적인 오류 재시도는 저렴할 수 있지만 제한 시간 초과 또는 제한 오류 재시도는 비용이 더 많이 들 수 있습니다.

시도가 성공하면 버킷에 용량이 반환됩니다. 버킷은 최대 용량을 초과하여 증가하거나 0 미만으로 감소할 수 없습니다.

`useCircuitBreakerMode` 설정 값에 따라가 용량을 0 미만으로 줄이면 다음 결과 중 하나가 발생합니다.

- 설정이 TRUE인 경우 예외가 발생합니다. 예를 들어 재시도가 너무 많아 재시도가 성공할 가능성이 낮은 경우가 있습니다.
- 설정이 FALSE인 경우 지연이 발생합니다. 예를 들어 버킷의 용량이 다시 충분해질 때까지 지연됩니다.

토큰 버킷 파라미터는 `tokenBucket DSL 블록`에서 구성할 수 있습니다.

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

재시도 토큰 버킷을 튜닝하는 데 사용할 수 있는 파라미터는 다음과 같습니다.

파라미터	기본값	설명
initialTryCost	0	초기 시도 시 버킷에서 감소하는 양입니다. 기본값인 0은 용량이 감소하지 않으므로 초기 시도가 중지되거나 지연되지 않음을 의미합니다.
initialTrySuccessIncrement	1	초기 시도가 성공했을 때 용량을 늘리는 양입니다.
maxCapacity	500	토큰 버킷의 최대 용량입니다. 사용 가능한 토큰 수는이 수를 초과할 수 없습니다.
refillUnitsPerSecond	0	초당 버킷에 다시 추가된 용량입니다. 값이 0이면 용량이 자동으로 다시 추가되지 않음을 의미합니다. (예: 성공한 시도만 용량을 증가시킵니다). 값이 0이면 TRUE여야 useCircuitBreakerMode 합니다.
retryCost	5	일시적인 실패 후 시도 시 버킷에서 감소하는 양입니다. 시도가 성공하면 동일한 양이 버킷으로 다시 증가합니다.
timeoutRetryCost	10	제한 시간 초과 또는 제한 실패 후 시도 시 버킷에서 감소하는 양입니다. 시도가 성공하면 동일한 양이 버킷으로 다시 증가합니다.
useCircuitBreakerMode	TRUE	용량을 줄이려고 할 때 버킷의 용량이 0 아래로 떨어질 때의 동작을 결정합니다. TRUE인 경우 토큰 버킷은 더 이상 재시

파라미터	기본값	설명
		도 용량이 존재하지 않음을 나타내는 예외를 발생시킵니다. FALSE일 때 토큰 버킷은 충분한 용량이 다시 채워질 때까지 시도를 지연시킵니다.

적응형 재시도

표준 재시도 전략의 대안으로 적응형 재시도 전략은 제한 오류를 최소화하기 위한 이상적인 요청 속도를 찾는 고급 접근 방식입니다.

> **Important**
>
> 적응형 재시도는 고급 재시도 모드입니다. 이 재시도 전략을 사용하는 것은 일반적으로 권장되지 않습니다.

적응형 재시도에는 표준 재시도의 모든 기능이 포함됩니다. 제한되지 않은 요청과 비교하여 제한되는 요청의 속도를 측정하는 클라이언트 측 속도 제한기를 추가합니다. 또한 트래픽을 제한하여 안전한 대역폭 내에 유지하려고 시도하므로 제한 오류가 발생하지 않는 것이 이상적입니다.

속도는 변화하는 서비스 조건 및 트래픽 패턴에 실시간으로 적응하며 그에 따라 트래픽 속도를 높이거나 낮출 수 있습니다. 매우 심각하게도 속도 제한기는 트래픽이 많은 시나리오에서 초기 시도를 지연시킬 수 있습니다.

`retryStrategy` 메서드에 추가 파라미터를 제공하여 적응형 재시도 전략을 선택합니다. 속도 제한기 파라미터는 `rateLimiter` DSL 블록에서 구성할 수 있습니다.

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

> **Note**
>
> 적응형 재시도 전략은 클라이언트가 단일 리소스(예: DynamoDB 테이블 하나 또는 Amazon S3 버킷 하나)에 대해 작동한다고 가정합니다.
> 단일 클라이언트를 여러 리소스에 사용하는 경우 한 리소스와 연결된 제한 또는 중단으로 인해 클라이언트가 다른 모든 리소스에 액세스할 때 지연 시간이 증가하고 장애가 발생합니다. 적응형 재시도 전략을 사용하는 경우 각 리소스에 대해 단일 클라이언트를 사용하는 것이 좋습니다.

관찰성

관찰성은 시스템의 현재 상태를 내보내는 데이터에서 추론할 수 있는 정도입니다. 방출되는 데이터를 일반적으로 원격 측정이라고 합니다.

는 지표, 트레이스 및 로그라는 세 가지 일반적인 원격 측정 신호를 모두 제공할 AWS SDK for Kotlin 수 있습니다. `TelemetryProvider`를 연결하여 원격 측정 데이터를 관찰성 백엔드(예: AWS X-Ray 또는 Amazon CloudWatch)로 전송한 다음 조치를 취할 수 있습니다.

기본적으로 로깅만 활성화되고 SDK에서 다른 원격 측정 신호는 비활성화됩니다. 이 주제에서는 원격 측정 출력을 활성화하고 구성하는 방법을 설명합니다.

> **Important**
>
> `TelemetryProvider`는 현재 사용하기 위해 옵트인해야 하는 실험 API입니다.

구성 `TelemetryProvider`

모든 서비스 클라이언트 또는 개별 클라이언트`TelemetryProvider`에 대해 애플리케이션에서를 전역적으로 구성할 수 있습니다. 다음 예제에서는 가상 함수를 사용하여 `TelemetryProvider` API 작업을 `getConfiguredProvider()` 보여줍니다. 이 the section called “원격 측정 공급자” 섹션에서는 SDK에서 제공하는 구현에 대한 정보를 설명합니다. 공급자가 지원되지 않는 경우 자체 지원을 구현하거나 GitHub에서 기능 요청을 열 수 있습니다.

기본 글로벌 원격 측정 공급자 구성

기본적으로 모든 서비스 클라이언트는 전역적으로 사용 가능한 원격 측정 공급자를 사용하려고 시도합니다. 이렇게 하면 공급자를 한 번 설정할 수 있으며 모든 클라이언트가 공급자를 사용합니다. 이 작업은 서비스 클라이언트를 인스턴스화하기 전에 한 번만 수행해야 합니다.

글로벌 원격 측정 공급자를 사용하려면 먼저 다음 Gradle 코드 조각과 같이 프로젝트 종속성을 업데이트하여 원격 측정 기본 모듈을 추가합니다.

(*X.Y.Z* 링크로 이동하여 사용 가능한 최신 버전을 볼 수 있습니다.)

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

그런 다음 다음 코드와 같이 서비스 클라이언트를 생성하기 전에 글로벌 원격 측정 공급자를 설정합니다.

```
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")
}
```

특정 서비스 클라이언트에 대한 원격 측정 공급자 구성

특정 원격 측정 공급자(글로벌 공급자 제외)를 사용하여 개별 서비스 클라이언트를 구성할 수 있습니다. 방법은 다음 예제와 같습니다.

```
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")
}
```

Metrics

다음 표에는 SDK가 내보내는 원격 측정 지표가 나열되어 있습니다. 지표를 관찰할 수 있도록 원격 측정 공급자를 구성합니다.

방출되는 지표는 무엇입니까?

메트릭 이름	단위	유형	속성	설명
smithy.client.call.duration	s	히스토그램(Histogram)	rpc.service, rpc.method	전체 통화 지속 시간(재시도 포함)
smithy.client.call.attempts	{시도}	MonotonCounter	rpc.service, rpc.method	개별 작업에 대한 시도 횟수
smithy.client.call.errors	{error}	MonotonCounter	rpc.service, rpc.method, exception.type	작업의 오류 수
smithy.client.call.attempt_duration	s	히스토그램(Histogram)	rpc.service, rpc.method	서비스에 연결하고, 요청을 보내고, HTTP 상태 코드 및 헤더를 다시 가져오는 데 걸리는 시간(전송 대기 시간 포함)

메트릭 이름	단위	유형	속성	설명
smithy.client.call.resolve_endpoint_duration	s	히스토그램(Histogram)	rpc.service, rpc.method	요청에 대한 엔드포인트(엔드포인트 해석기, DNS 아님)를 해결하는 데 걸리는 시간
smithy.client.call.serialization_duration	s	히스토그램(Histogram)	rpc.service, rpc.method	메시지 본문을 직렬화하는 데 걸리는 시간
smithy.client.call.deserialization_duration	s	히스토그램(Histogram)	rpc.service, rpc.method	메시지 본문을 역직렬화하는 데 걸리는 시간
smithy.client.call.auth.signing_duration	s	히스토그램(Histogram)	rpc.service, rpc.method, auth.scheme_id	요청에 서명하는 데 걸리는 시간
smithy.client.call.auth.resolve_identity_duration	s	히스토그램(Histogram)	rpc.service, rpc.method, auth.scheme_id	자격 증명 공급자로부터 자격 증명(예: AWS 자격 증명 또는 보유자 토큰)을 획득하는 데 걸리는 시간
smithy.client.http.connections.acquire_duration	s	히스토그램(Histogram)		요청을 통해 연결을 획득하는 데 걸리는 시간
smithy.client.http.connections.limit	{connection}	[비동기]UpDownCount		HTTP 클라이언트에 대해 허용/구성된 최대 열린 연결 수

메트릭 이름	단위	유형	속성	설명
smithy.client.http.connections.usage	{connection}	[비동기]UpDownCount	상태: 유휴 | 획득	연결 풀의 현재 상태
smithy.client.http.connections.uptime	s	히스토그램(Histogram)		연결이 열린 시간
smithy.client.http.requests.usage	{요청}	[비동기]UpDownCount	상태: 대기열에 있음 | 진행 중	HTTP 클라이언트 요청 동시성의 현재 상태
smithy.client.http.requests.queued_duration	s	히스토그램(Histogram)		HTTP 클라이언트가 요청을 실행하기 위해 대기하고 대기한 시간
smithy.client.http.bytes_sent	By	MonotonCounter	server.address	HTTP 클라이언트가 보낸 총 바이트 수
smithy.client.http.bytes_received	By	MonotonCounter	server.address	HTTP 클라이언트가 수신한 총 바이트 수

다음은 열 설명입니다.

- 지표 이름 - 내보낸 지표의 이름입니다.
- 단위 - 지표의 측정 단위입니다. 단위는 UCUM 대/소문자 구분("c/s") 표기법으로 제공됩니다.
- 유형 - 지표를 캡처하는 데 사용되는 계측의 유형입니다.
- 설명 - 지표가 측정하는 항목에 대한 설명입니다.
- 속성 - 지표와 함께 방출되는 속성(차원)의 집합입니다.

로깅

는 SLF4J 호환 로거를 원격 측정 공급자LoggerProvider의 기본값으로 AWS SDK for Kotlin 구성합니다. 추상화 계층인 SLF4J를 사용하면 런타임에 여러 로깅 시스템 중 하나를 사용할 수 있습니다. 지원되는 로깅 시스템에는 Java 로깅 APIs, Log4j 2 및 Logback이 포함됩니다.

> **Warning**
>
> 디버깅 용도로만 유선 로깅을 사용하는 것이 좋습니다. (전신 로깅은 아래에서 설명합니다.) 프로덕션 환경에서는 이메일 주소, 보안 토큰, API 키, 암호 및 AWS Secrets Manager 보안 암호와 같은 민감한 데이터를 로깅할 수 있으므로 끕니다. 와이어 로깅은 HTTPS 호출의 경우에도 암호화 없이 전체 요청 또는 응답을 로깅합니다.
> 대규모 요청(예: Amazon S3에 파일 업로드) 또는 응답의 경우 상세 유선 로깅도 애플리케이션의 성능에 큰 영향을 미칠 수 있습니다.

예제 Log4j 2 로깅 구성

SLF4J호환되는 로그 라이브러리를 사용할 수 있지만이 예제에서는 Log4j 2를 사용하여 JVM 프로그램의 SDK에서 로그 출력을 활성화합니다.

그래들 종속성

(*X.Y.Z* 링크로 이동하여 사용 가능한 최신 버전을 볼 수 있습니다.)

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

Log4j 2 구성 파일

resources 디렉터리log4j2.xml에 라는 파일을 생성합니다(예: <project-dir>/src/main/resources). 파일에 다음 XML 구성을 추가합니다.

```
<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>
```

이 구성에는 MDC(맵핑된 진단 컨텍스트) 로깅을 활성화하는 pattern 속성의 %X 지정자가 포함됩니다.

SDK는 각 작업에 대해 다음 MDC 요소를 추가합니다.

rpc

호출된 RPC의 이름입니다. 예: S3.GetObject.

sdkInvocationId

작업에 대해 서비스 클라이언트가 할당한 고유 ID입니다. ID는 단일 작업의 호출과 관련된 모든 로깅 이벤트의 상관 관계를 나타냅니다.

와이어 수준 메시지에 대한 로그 모드 지정

기본적으로는 API 요청 및 응답의 민감한 데이터를 포함할 수 있으므로 와이어 수준 메시지를 로깅하지 AWS SDK for Kotlin 않습니다. 그러나 디버깅을 위해 이러한 수준의 세부 정보가 필요한 경우가 있습니다.

Kotlin SDK를 사용하면 코드에서 로그 모드를 설정하거나 환경 설정을 사용하여 다음에 대한 디버그 메시징을 활성화할 수 있습니다.

- HTTP 요청
- HTTP 응답

로그 모드는 각 비트가 플래그(모드)이고 값이 추가되는 비트 필드로 지원됩니다. 하나의 요청 모드와 하나의 응답 모드를 결합할 수 있습니다.

코드에서 로그 모드 설정

추가 로깅을 옵트인하려면 서비스 클라이언트를 구성할 때 logMode 속성을 설정합니다.

다음 예제에서는 요청(본체 사용) 및 응답(본체 미사용)의 로깅을 활성화하는 방법을 보여줍니다.

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

// ...

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

서비스 클라이언트 구성 중에 설정된 로그 모드 값은 환경에서 설정된 모든 로그 모드 값을 재정의합니다.

환경에서 로그 모드 설정

코드에 명시적으로 구성되지 않은 모든 서비스 클라이언트에 대해 전역적으로 로그 모드를 설정하려면 다음 중 하나를 사용합니다.

- JVM 시스템 속성: `sdk.logMode`
- 환경 변수: `SDK_LOG_MODE`

다음과 같은 대/소문자를 구분하지 않는 값을 사용할 수 있습니다.

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

환경의 설정을 사용하여 결합된 로그 모드를 생성하려면 값을 파이프(`|`) 기호로 구분합니다.

예를 들어 다음 예제에서는 이전 예제와 동일한 로그 모드를 설정합니다.

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

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

> **Note**
>
> 또한 호환되는 SLF4J 로거를 구성하고 로깅 수준을 DEBUG로 설정하여 와이어 수준 로깅을 활성화해야 합니다.

원격 측정 공급자

SDK는 현재 공급자로서 OpenTelemetry(OTel)를 지원합니다. SDK는 향후 추가 원격 측정 공급자를 제공할 수 있습니다.

주제

- OpenTelemetry 기반 원격 측정 공급자 구성

OpenTelemetry 기반 원격 측정 공급자 구성

SDK for Kotlin은 OpenTelemetry에서 지원하는 `TelemetryProvider` 인터페이스의 구현을 제공합니다.

사전 조건

다음 Gradle 코드 조각과 같이 OpenTelemetry 공급자를 추가하도록 프로젝트 종속성을 업데이트합니다. *X.Y.Z* 링크로 이동하여 사용 가능한 최신 버전을 확인할 수 있습니다.

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

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

SDK 구성

다음 코드는 OpenTelemetry 원격 측정 공급자를 사용하여 서비스 클라이언트를 구성합니다.

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

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

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

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

> **Note**
>
> OpenTelemetry SDK를 구성하는 방법에 대한 설명은이 가이드의 범위를 벗어납니다. OpenTelemetry Java 설명서에는 수동, Java 에이전트를 통한 자동 또는 (선택 사항) 수집기 등 다양한 접근 방식에 대한 구성 정보가 포함되어 있습니다.

리소스

OpenTelemetry를 시작하는 데 도움이 되는 다음 리소스를 사용할 수 있습니다.

- AWS Distro for OpenTelemetry - AWS OTeL Distro 홈페이지
- aws-otel-java-instrumentation - AWS Distro for OpenTelemetry Java Instrumentation Library
- aws-otel-lambda AWS 관리형 OpenTelemetry Lambda 계층
- aws-otel-collector - AWS Distro for OpenTelemetry Collector
- AWS 관찰성 모범 사례 -와 관련된 관찰성에 대한 일반적인 모범 사례 AWS

서비스 클라이언트 구성 재정의

서비스 클라이언트가 생성되면 서비스 클라이언트는 모든 작업에 고정 구성을 사용합니다. 하지만 하나 이상의 특정 작업에 대해 구성을 재정의해야 하는 경우가 있습니다.

각 서비스 클라이언트에는 기존 구성의 복사본을 수정할 수 있도록 `withConfig` 확장이 있습니다. `withConfig` 확장은 수정된 구성의 새 서비스 클라이언트를 반환합니다. 원래 클라이언트는 독립적으로 존재하며 원래 구성을 사용합니다.

다음 예제는 두 작업을 호출하는 S3Client 인스턴스의 생성을 보여줍니다.

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

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

다음 코드 조각은 단일 listObjectV2 작업에 대한 구성을 재정의하는 방법을 보여줍니다.

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

s3 클라이언트에 대한 작업 호출은 클라이언트가 생성될 때 지정된 원래 구성을 사용합니다. 구성에는 리전에 us-west-2 region 대한 요청 로깅 및가 포함됩니다.

overriddenS3 클라이언트의 listObjectsV2 호출은 현재 인 리전을 제외하고 원래 s3 클라이언트와 동일한 설정을 사용합니다eu-central-1.

재정의된 클라이언트의 수명 주기

이전 예제에서 s3 클라이언트와 overriddenS3 클라이언트는 서로 독립적입니다. 작업은 열린 상태로 유지되는 한 클라이언트에서 호출할 수 있습니다. 각는 별도의 구성을 사용하지만 기본 리소스(예: HTTP 엔진)도 재정의되지 않는 한 공유할 수 있습니다.

재정의된 구성으로 클라이언트를 닫고 원래 클라이언트를 별도로 닫습니다. 원래 클라이언트를 닫기 전후에 재정의된 구성으로 클라이언트를 닫을 수 있습니다. 구성이 재정의된 클라이언트를 장기간 사용해야 하는 경우가 아니면 use 메서드로 수명 주기를 래핑하는 것이 좋습니다. use 메서드는 예외가 발생할 경우 클라이언트가 닫히도록 합니다.

클라이언트 간에 공유되는 리소스

를 사용하여 서비스 클라이언트를 생성할 때 원래 클라이언트와 리소스를 공유할 withConfig수 있습니다. 반대로 fromEnvironment를 사용하여 클라이언트를 생성하거나 명시적으로 구성하면 클라이

언트는 독립 리소스를 사용합니다. HTTP 엔진 및 자격 증명 공급자와 같은 리소스는 `withConfig` 블록에서 재정의되지 않는 한 공유됩니다.

각 클라이언트의 수명 주기는 독립적이므로 공유 리소스는 마지막 클라이언트가 닫힐 때까지 열려 있고 사용할 수 있습니다. 따라서 더 이상 필요하지 않은 경우 재정의된 서비스 클라이언트를 닫는 것이 중요합니다. 이렇게 하면 공유 리소스가 열린 상태로 유지되고 메모리, 연결 및 CPU 주기와 같은 시스템 리소스가 소비되지 않습니다.

다음 예제에서는 공유 리소스와 독립 리소스를 모두 보여줍니다.

`s3` 및 `overriddenS3` 클라이언트는 캐싱 구성을 포함하여 동일한 자격 증명 공급자 인스턴스를 공유합니다. 캐시된 값이 `s3` 클라이언트의 호출에서 여전히 최신 상태인 경우에서 수행한 호출은 자격 증명을 `overriddenS3` 재사용합니다.

HTTP 엔진은 두 클라이언트 간에 공유되지 않습니다. 각 클라이언트에는 `withConfig` 호출에서 재정의되었기 때문에 독립 HTTP 엔진이 있습니다.

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

s3.listBuckets { ... }

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

SDK 사용

이 섹션에서는를 사용하는 데 필요한 기본 정보를 제공합니다 AWS SDK for Kotlin.

주제

- 요청을 생성
- 코루틴
- 스트리밍 작업
- 페이지 매김
- Waiters
- 오류 처리
- 사전 서명 요청
- FAQs 문제 해결

요청을 생성

서비스 클라이언트를 사용하여에 요청합니다 AWS 서비스. 는 형식 안전 빌더 패턴에 따라 요청을 생성하는 도메인별 언어(DSLs)를 AWS SDK for Kotlin 제공합니다. 중첩된 요청 구조도를 통해 액세스할 수 있습니다DSLs.

다음 예제에서는 Amazon DynamoDB createTable 작업 입력을 생성하는 방법을 보여줍니다.

```
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)
```

서비스 인터페이스 DSL 과부하

서비스 클라이언트 인터페이스의 각 비스트리밍 작업에는 DSL 오버로드가 있으므로 별도의 요청을 생성할 필요가 없습니다.

오버로드된 함수를 사용하여 Amazon Simple Storage Service(Amazon S3) 버킷을 생성하는 예:

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

이는 다음과 동일합니다.

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

s3client.createBucket(request)
```

필수 입력이 없는 요청

필수 입력이 없는 작업은 요청 객체를 전달할 필요 없이 호출할 수 있습니다. 이는 Amazon S3 작업과 같은 목록 유형 `listBuckets` API 작업에서 종종 가능합니다.

예를 들어 다음 세 문은 동일합니다.

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

코루틴

는 기본적으로 비동기식 AWS SDK for Kotlin 입니다. KotlinSDK용는 모든 작업에 suspend 함수를 사용하며, 이는 코루틴에서 호출하기 위한 것입니다.

코루틴에 대한 자세한 지침은 공식 Kotlin 설명서를 참조하세요.

동시 요청

비동기 코루틴 빌더를 사용하여 결과에 신경을 쓰는 동시 요청을 시작할 수 있습니다.는 나중에 결과를 제공하겠다는 약속을 나타내는 가볍고 차단되지 않는 미래를 나타내는 지연된를 async 반환합니다.

결과에 신경 쓰지 않는 경우(작업이 완료된 경우에만 해당) 시작 코루틴 빌더를 사용할 수 있습니다. `launch`는 개념적으로와 유사합니다`async`. 차이점은 시작이 작업을 반환하고 결과 값을 포함하지 않는 반면는를 async 반환한다는 것입니다`Deferred`.

다음은 두 키의 콘텐츠 크기를 가져오기 위해 headObject 작업을 사용하여 Amazon S3에 동시 요청하는 예제입니다.

```
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")

}
```

차단 요청

코루틴을 사용하지 않고 다른 스레딩 모델을 구현하는 기존 코드에서 서비스를 호출하려면 코루틴 runBlocking 빌더를 사용할 수 있습니다. 다른 스레딩 모델의 예는 Java의 기존 실행기/미래 접근 방식을 사용하는 것입니다. Java 및 Kotlin 코드 또는 라이브러리를 블렌딩하는 경우이 접근 방식을 사용해야 할 수 있습니다.

이름에서 알 수 있듯이이 `runBlocking`빌더는 새 코루틴을 시작하고 완료될 때까지 현재 스레드를 차단합니다.

> **Warning**
>
> `runBlocking`는 일반적으로 코루틴에서 사용해서는 안 됩니다. 일반 차단 코드를 일시 중지 스타일(예: 기본 함수 및 테스트)로 작성된 라이브러리에 브리지하도록 설계되었습니다.

스트리밍 작업

에서 AWS SDK for Kotlin이진 데이터(스트림)는 추상적인 읽기 전용 바이트 스트림인 `ByteStream` 유형으로 표시됩니다.

스트리밍 응답

바이너리 스트림(예: Amazon Simple Storage Service(Amazon S3) GetObject API 작업)을 사용한 응답은 다른 방법과 다르게 처리됩니다. 이러한 메서드는 응답을 직접 반환하는 대신 응답을 처리하는 lambda 함수를 사용합니다. 이렇게 하면 함수에 대한 응답 범위가 제한되고 호출자와 SDK 런타임 모두에 대한 수명 관리가 간소화됩니다.

lambda 함수가 반환되면 기본 HTTP 연결과 같은 모든 리소스가 릴리스됩니다. (lambda가 반환`ByteStream`된 후에는에 액세스해서는 안 되며 종료에서 전달해서는 안 됩니다.) 호출 결과는 lambda가 반환하는 모든 값입니다.

다음 코드 예제는 응답을 처리하는 lambda 파라미터를 수신하는 getObject 함수를 보여줍니다.

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

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

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

val contentSize = s3Client.getObject(req) { resp ->
    // resp is valid until the end of the block.
    // Do not attempt to store or process the stream after the block returns.

    // resp.body is of type ByteStream.
    val rc = resp.body?.writeToFile(path)
    rc
```

```
}
println("wrote $contentSize bytes to $path")
```

ByteStream 유형에는 일반적인 사용 방법에 대한 다음과 같은 확장이 있습니다.

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

이 모든 것은 aws.smithy.kotlin.runtime.content 패키지에 정의되어 있습니다.

스트리밍 요청

를 제공하기 위해 다음을 포함한 몇 ByteStream가지 편의 방법도 있습니다.

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

이 모든 것은 aws.smithy.kotlin.runtime.content 패키지에 정의되어 있습니다.

다음 코드 예제는 PutObjectRequest를 생성할 때 본문 속성을 제공하는 ByteStream 편의 방법의 사용을 보여줍니다.

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

페이지 매김

많은 AWS 작업은 페이로드가 너무 커서 단일 응답으로 반환할 수 없는 경우 페이지가 매겨진 결과를 반환합니다. 에는 결과를 자동으로 페이지 매김하는 서비스 클라이언트 인터페이스에 대한 확장이 AWS SDK for Kotlin 포함되어 있습니다. 결과를 처리하는 코드만 작성하면 됩니다.

페이지 매김은 Flow<T>로 노출되므로 비동기 컬렉션(예: `map`, `filter`및)에 대한 Kotlin의 관용 변환을 활용할 수 있습니다`take`. 예외는 투명하므로 오류 처리가 일반 API 호출처럼 느껴지며 취소는 일반적인 코루틴 협동 취소를 따릅니다. 자세한 내용은 공식 가이드의 흐름 및 흐름 예외를 참조하세요.

> **Note**
>
> 다음 예제에서는 Amazon S3를 사용합니다. 그러나 하나 이상의 페이지가 매겨진가 있는 모든 서비스에서 개념은 동일합니다APIs. 모든 페이지 매김 확장은 `aws.sdk.kotlin.<service>.paginators` 패키지에 정의됩니다(예: `aws.sdk.kotlin.dynamodb.paginators`).

다음 코드 예제는 listObjectsV2Paginated 함수 호출에서 페이지가 지정된 응답을 처리하는 방법을 보여줍니다.

가져오기

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

코드

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

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

Waiters

웨이터는 원하는 상태에 도달할 때까지 또는 리소스가 원하는 상태로 전환되지 않을 것으로 확인될 때까지 리소스를 폴링하는 데 사용되는 클라이언트 측 추상화입니다. 이는 Amazon Simple Storage Service(Amazon S3)와 같이 최종적으로 일관된 서비스 또는 Amazon와 같이 비동기적으로 리소스를 생성하는 서비스를 사용할 때 일반적으로 수행하는 작업입니다EC2.

리소스의 상태를 지속적으로 폴링하기 위해 로직을 작성하는 것은 번거롭고 오류가 발생하기 쉽습니다. 웨이터의 목표는이 책임을 고객 코드에서 로 옮기는 것입니다. AWS SDK for Kotlin는 AWS 작업의 타이밍 측면에 대한 심층적인 지식을 갖추고 있습니다.

> **Note**
>
> 다음 예제에서는 Amazon S3를 사용합니다. 그러나 개념은 하나 이상의 웨이터가 정의된 모든 AWS 서비스 에 대해 동일합니다. 모든 확장은 `aws.sdk.kotlin.<service>.waiters` 패키지에 정의됩니다(예: `aws.sdk.kotlin.dynamodb.waiters`). 또한 표준 명명 규칙()을 따릅니다`waitUntil<Condition>`.

다음 코드 예제는 폴링 로직을 쓰지 않도록 허용하는 웨이터 함수의 사용을 보여줍니다.

가져오기

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

코드

```
val s3 = S3Client.fromEnvironment()

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

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

// The bucket now exists.
```

> **Note**
>
> 각 대기 메서드는 원하는 조건에 도달하는 데 해당하는 최종 응답을 가져오는 데 사용할 수 있는 `Outcome` 인스턴스를 반환합니다. 결과에는 원하는 상태에 도달하기 위한 시도 횟수와 같은 추가 세부 정보도 포함됩니다.

오류 처리

에서 예외를 AWS SDK for Kotlin 발생시키는 방법과 시기를 이해하는 것은를 사용하여 고품질 애플리케이션을 구축하는 데 중요합니다SDK. 다음 섹션에서는에서 발생하는 다양한 예외 사례SDK와 이를 적절하게 처리하는 방법을 설명합니다.

서비스 예외

가장 일반적인 예외는 모든 서비스별 예외(예: `S3Exception`)가 상속`AwsServiceException`하는 입니다. 이 예외는 AWS 서비스의 오류 응답을 나타냅니다. 예를 들어 존재하지 않는 Amazon EC2 인스턴스를 종료하려고 하면 Amazon은 오류 응답을 EC2 반환합니다. 오류 응답 세부 정보는 `AwsServiceException` 발생한에 포함됩니다.

가 발생하면 요청이 로 성공적으로 전송되었지만 처리할 수 AWS 서비스 없음을 `AwsServiceException`의미합니다. 이는 요청의 파라미터 오류 또는 서비스 측의 문제로 인해 발생할 수 있습니다.

클라이언트 예외

`ClientException`는 요청을 보내려고 하거나 응답을 구문 분석하려고 할 AWS 때 클라이언트 코드 내에서 AWS SDK for Kotlin 문제가 발생했음을 나타냅니다 AWS. `ClientException`는 일반적으로 보다 심각`AwsServiceException`하며 클라이언트가 서비스 호출을 처리하지 못하는 주요 문제를 나타냅니다 AWS 서비스. 예를 들어는 서비스의 응답을 구문 분석하지 `ClientException` 못하면를 AWS SDK for Kotlin 발생시킵니다.

오류 메타데이터

모든 서비스 예외 및 클라이언트 예외에는 `sdkErrorMetadata` 속성이 있습니다. 오류에 대한 추가 세부 정보를 검색하는 데 사용할 수 있는 유형 속성 백입니다.

다음을 포함하되 이에 국한되지 않는 몇 가지 사전 정의된 확장이 `AwsErrorMetadata` 유형에 직접 존재합니다.

- `sdkErrorMetadata.requestId` - 고유 요청 ID
- `sdkErrorMetadata.errorMessage` - 사람이 읽을 수 있는 메시지(일반적으로와 일치하지`Exception.message`만 서비스에 예외가 알려지지 않은 경우 추가 정보가 포함될 수 있음)
- `sdkErrorMetadata.protocolResponse` - 원시 프로토콜 응답

다음 예제에서는 오류 메타데이터에 액세스하는 방법을 보여줍니다.

```
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}")
}
```

사전 서명 요청

나중에 다른 호출자가 자신의 자격 증명을 제시하지 않고 요청을 사용할 수 있도록 일부 AWS API 작업에 대한 요청을 미리 서명할 수 있습니다.

예를 들어 Alice가 Amazon Simple Storage Service(Amazon S3) 객체에 액세스할 수 있고 Bob과 객체 액세스를 일시적으로 공유하려고 한다고 가정합니다. Alice는 미리 서명된 `GetObject` 요청을 생성하여 Bob과 공유하므로 Alice의 자격 증명에 액세스할 필요 없이 객체를 다운로드할 수 있습니다.

기본 사항 사전 지정

SDK for Kotlin은 서비스 클라이언트에서 요청에 사전 서명할 수 있는 확장 방법을 제공합니다. 미리 서명된 모든 요청에는 서명된 요청의 유효 기간을 나타내는 기간이 필요합니다. 기간이 끝나면 미리 서명된 요청이 만료되고 실행되면 인증 오류가 발생합니다.

다음 코드는 Amazon S3에 대해 미리 서명된 `GetObject` 요청을 생성하는 예제를 보여줍니다. 요청은 생성 후 24시간 동안 유효합니다.

```
val s3 = S3Client.fromEnvironment()

val unsignedRequest = GetObjectRequest {
```

```
    bucket = "foo"
    key = "bar"
}

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

presignGetObject 확장 메서드는 HttpRequest 객체를 반환합니다. 요청 객체에는 작업을 호출할 수 URL 있는 미리 서명된이 포함되어 있습니다. 다른 호출자는 다른 코드베이스 또는 프로그래밍 언어 환경에서 URL (또는 전체 요청)을 사용할 수 있습니다.

미리 서명된 요청을 생성한 후 HTTP 클라이언트를 사용하여 요청을 호출합니다. HTTP GET 요청을 호출API하는는 HTTP 클라이언트에 따라 다릅니다. 다음 예제에서는 Kotlin URL.readText 메서드를 사용합니다.

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

고급 사전 서명 구성

에서 요청을 미리 서명할 수 있는 SDK각 메서드에는 특정 서명자 구현 또는 세부 서명 파라미터와 같은 고급 구성 옵션을 제공하는 데 사용할 수 있는 과부하가 있습니다.

다음 예제에서는 CRT 서명자 변형을 사용하고 향후 서명 날짜를 지정하는 Amazon S3 GetObject 요청을 보여줍니다.

```
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
}
```

반환된 미리 서명된 요청은 24시간 전 날짜이며 그 이전에는 유효하지 않습니다. 그 후 8시간이 지나면 만료됩니다.

사전 서명 POST 및 PUT 요청

사전 서명이 가능한 많은 작업에는 URL 만 필요하며 HTTP GET 요청으로 실행해야 합니다. 그러나 일부 작업은 본문을 사용하므로 경우에 따라 헤더와 함께 HTTP POST 또는 HTTP PUT 요청으로 실행해야 합니다. 이러한 요청의 사전 서명은 사전 서명 GET 요청과 동일하지만 미리 서명된 요청을 호출하는 것은 더 복잡합니다.

다음은 S3 `PutObject` 요청에 사전 서명하는 예제입니다.

```
val s3 = S3Client.fromEnvironment()

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

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

반환된의 메`HttpRequest`서드 값은 `HttpMethod.PUT` 이며, 향후 HTTP 요청 호출에 포함되어야 하는 URL 및 헤더가 포함됩니다. 이 요청을 다른 코드베이스 또는 프로그래밍 언어 환경에서 실행할 수 있는 호출자에게 전달할 수 있습니다.

미리 서명된 POST 또는 PUT 요청을 생성한 후 HTTP 클라이언트를 사용하여 요청을 호출합니다. POST 또는 PUT 요청을 호출API하는는 사용된 HTTP 클라이언트에 URL 따라 다릅니다. 다음 예제에서는 OkHttp HTTP 클라이언트를 사용하고를 포함하는 본문을 포함합니다`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()
```

가 미리 서명SDK할 수 있는 작업

KotlinSDK용는 현재 연결된 HTTP 메서드를 사용하여 호출해야 하는 다음 API 작업의 사전 서명을 지원합니다.

AWS 서비스	Operation	SDK 확장 메서드	HTTP 메서드 사용
Amazon S3	GetObject	presignGetObject	HTTP GET
Amazon S3	PutObject	presignPutObject	HTTP PUT
Amazon S3	UploadPart	presignUploadPart	HTTP PUT
AWS Security Token Service	GetCallerIdentity	presignGetCaller자격 증명	HTTP POST
Amazon Polly	SynthesizeSpeech	presignSynthesizeSpeech	HTTP POST

FAQs 문제 해결

애플리케이션에서 AWS SDK for Kotlin 를 사용하면이 주제에 나열된 몇 가지 문제가 발생할 수 있습니다. 다음 제안을 사용하여 근본 원인을 파악하고 오류를 해결합니다.

“연결 종료” 문제를 해결하려면 어떻게 해야 합니까?

다음 유형 중 하나와 같은 예외로 '연결 닫힘' 문제가 발생할 수 있습니다.

- `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)`

이러한 예외는에서 서비스로SDK의 TCP 연결이 예기치 않게 닫히거나 재설정되었음을 나타냅니다. 호스트, AWS 서비스 또는 NAT 게이트웨이, 프록시 또는 로드 밸런서와 같은 중개자가 연결을 종료했을 수 있습니다.

이러한 유형의 예외는 자동으로 재시도되지만 로깅 구성에 따라 SDK 로그에 계속 표시될 수 있습니다. 코드에 예외가 발생하는 경우, 이는 활성 재시도 전략이 최대 시도 또는 재시도 토큰 버킷과 같이 구성된 한도를 소진했음을 나타냅니다. 재시도 전략에 대한 자세한 내용은이 가이드의 the section called "재시도" 섹션을 참조하세요. the section called "최대 시도 횟수에 도달하기 전에 예외가 발생하는 이유는 무엇입니까?" 주제도 참조하세요.

최대 시도 횟수에 도달하기 전에 예외가 발생하는 이유는 무엇입니까?

경우에 따라 재시도할 것으로 예상되었지만 대신 발생했던 예외가 표시될 수 있습니다. 이러한 상황에서는 다음 단계가 문제를 해결하는 데 도움이 될 수 있습니다.

- 예외를 재시도할 수 있는지 확인합니다. 예를 들어 잘못된 서비스 요청, 권한 부족 및 존재하지 않는 리소스를 나타내는 예외와 같은 일부 예외는 재시도할 수 없습니다. SDK는 이러한 종류의 예외를 자동으로 재시도하지 않습니다. 에서 상속되는 예외를 포착하는 경우 부울 속성을 확인하여 SDK가 예외를 재시도할 수 있다고 판단했는지 `SdkBaseException.sdkErrorMetadata.isRetryable` 확인할 `SdkBaseException`수 있습니다.
- 예외가 코드에 발생하고 있는지 확인합니다. 일부 예외는 로그 메시지에 정보로 표시되지만 실제로 코드에 포함되지는 않습니다. 예를 들어가 여러 backoff-and-retry 주기를 통해 SDK 자동으로 작동하므로 제한 오류와 같은 재시도 가능한 예외가 기록될 수 있습니다. SDK 작업 호출은 구성된 재시도 설정에서 처리하지 않은 경우에만 예외가 발생합니다.
- 구성된 재시도 설정을 확인합니다. 재시도 전략 및 재시도 정책에 대한 자세한 내용은이 가이드의 the section called "재시도" 섹션을 참조하세요. 코드가 예상한 설정 또는 자동 기본값을 사용하고 있는지 확인합니다.
- 재시도 설정을 조정하는 것이 좋습니다. 이전 항목을 확인했지만 문제가 해결되지 않은 경우 재시도 설정을 조정하는 것이 좋습니다.
  - 최대 시도 횟수를 늘립니다. 기본적으로 작업에 대한 최대 시도 횟수는 3회입니다. 이것이 충분하지 않고 기본 설정에서 여전히 예외가 발생하는 경우 클라이언트 구성에서 `retryStrategy.maxAttempts` 속성을 늘리는 것이 좋습니다. 자세한 내용은 the section called "최대 시도 횟수" 섹션을 참조하세요.
  - 지연 설정을 늘립니다. 일부 예외는 기본 조건이 해결되기 전에 너무 빨리 재시도될 수 있습니다. 가 문제가 된다고 의심되는 경우 클라이언트 구성에서 `retryStrategy.delayProvider.initialDelay` 또는 `retryStrategy.delayProvider.maxBackoff` 속성을 늘리는 것이 좋습니다. 자세한 내용은 the section called "지연 및 백오프" 섹션을 참조하세요.

- 회로 차단기 모드를 비활성화합니다. 는 기본적으로 각 서비스 클라이언트에 대한 토큰 버킷을 SDK 유지합니다. 가 요청을 SDK 시도하고 재시도 가능한 예외로 실패하면 토큰 수가 감소하고, 요청이 성공하면 토큰 수가 증가합니다.

  기본적으로이 토큰 버킷이 남은 토큰이 0개에 도달하면 회로가 끊어집니다. 회로가 끊어지면는 재시도와 첫 번째 시도에서 실패한 현재 및 후속 요청을 SDK 즉시 예외로 간주합니다. 는 초기 시도가 성공하면 토큰 버킷에 충분한 용량을 반환한 후 SDK 재시도를 활성화합니다. 이 동작은 의도적인 것이며 서비스 중단 및 서비스 복구 중에 재시도 폭풍을 방지하도록 설계되었습니다.

  가 구성된 최대 시도까지 SDK 계속 재시도를 하려면 클라이언트 구성에서 `retryStrategy.tokenBucket.useCircuitBreakerMode` 속성을 false로 설정하여 회로 차단기 모드를 비활성화하는 것이 좋습니다. 이 속성을 false로 설정하면 SDK 클라이언트는 토큰 버킷이 추가 재시도를 중단하지 않고 충분한 용량에 도달할 때까지 대기하며, 이로 인해 토큰이 0개 남았을 때 예외가 발생할 수 있습니다.

`NoSuchMethodError` 또는를 수정하려면 어떻게 해야 합니까 NoClassDefFoundError?

SDK는 다양한 AWS 타사 종속성을 사용하여 올바르게 작동합니다. 예상 종속성이 런타임에 없거나 예상치 못한 버전인 경우 `NoSuchMethodError` 런타임 예외가 표시될 수 있습니다.

종속성 충돌은 일반적으로 SDK/Smithy 종속성 충돌과 서드 파티 종속성 충돌의 두 가지 범주로 나뉩니다.

Kotlin 애플리케이션을 빌드할 때 일반적으로 Gradle을 사용하여 종속성을 관리합니다. 애플리케이션에 SDK 서비스 클라이언트에 대한 종속성을 추가하면 자동으로 해결되고 모든 전이적 종속성이 포함됩니다. 애플리케이션에 다른 종속성이 있는 경우에 필요한 종속성과 충돌할 수 있습니다SDK(예: OkHttp 가 SDK 종속되는 일반적으로 사용되는 HTTP 클라이언트).

이러한 문제를 해결하려면 충돌을 방지하기 위해 특정 종속성 버전 또는 섀도우 종속성을 로컬 네임스페이스로 명시적으로 해결해야 할 수 있습니다. 그래들 종속성 해결은 그레이드 사용 설명서의 다음 섹션에서 설명하는 복잡한 주제입니다.

- 종속성 해결 이해
- 종속성 제약 조건 및 충돌 해결
- 종속성 버전 정렬

SDK/Smithy 종속성 충돌

일반적으로 SDK의 모듈은 버전 번호가 동일한 다른 SDK 모듈에 의존합니다. 예를 들어, `aws.sdk.kotlin:s3:1.2.3`는에 따라 다르고`aws.sdk.kotlin:aws-http:1.2.3`,는에 따라 다르며`aws.sdk.kotlin:aws-core:1.2.3`,는에 따라 달라집니다.

또한 SDK 모듈은 특정 통합 Smithy 모듈 버전도 사용합니다. 이러한 Smithy 버전 번호는 SDK 버전 번호와 동기화되지 않지만에서 예상하는 버전과 여전히 일치해야 합니다SDK. 예를 들어, `aws.sdk.kotlin:s3:1.2.3`는에 따라 달라지며`aws.smithy.kotlin:serde:1.1.1`,는 `aws.smithy.kotlin:runtime-core:1.1.1`에 따라 달라집니다.

이러한 버전 번호가 일치하지 않으면 종속성 충돌이 발생할 수 있습니다. 모든 SDK 종속성을 통합으로 업그레이드하고 명시적인 Smithy 종속성도 통합으로 업그레이드해야 합니다. Gradle 버전 카탈로그를 사용하여 버전을 동기화하고 및 Smithy 버전 간의 추측 매핑SDK을 제거하는 것이 좋습니다. 자세한 내용과 예제는 the section called “프로젝트 빌드 파일 생성” 주제를 참조하세요.

또한 SDK/Smithy 모듈의 마이너 버전 범프에는 SDK의 버전 관리 정책에 설명된 대로 중단되는 변경 사항이 포함될 수 있습니다. 마이너 버전 간에 업그레이드할 때는 변경 로그를 검사하고 런타임 동작을 철저히 검증하도록 각별히 주의해야 합니다.

에 **`NoClassDefFoundError`** 대한가 표시됩니다. **`okhttp3/coroutines/ExecuteAsyncKt`**

이 오류가 표시되면를 사용하도록 서비스 클라이언트를 구성하지 않은 것일 가능성이 높습니다`OkHttp4Engine`. Gradle을 구성하고 코드`OkHttp4Engine`에서를 사용하는 방법에 대한 설명서를 검토합니다.

를 AWS 서비스 사용하여 작업 AWS SDK for Kotlin

이 장에는 for Kotlin을 사용하여 AWS 서비스 를 사용하는 방법에 SDK 대한 정보가 포함되어 있습니다.

목차

- 를 사용하여 Amazon S3 작업 AWS SDK for Kotlin
  - 체크섬을 통한 데이터 무결성 보호
    - 객체 업로드
      - 미리 계산된 체크섬 값 사용
      - 멀티파트 업로드
    - 객체 다운로드
      - 비동기식 검증
  - KotlinSDK용를 사용하여 Amazon S3 다중 리전 액세스 포인트 작업
    - 다중 리전 액세스 포인트 작업
    - 객체 및 다중 리전 액세스 포인트 작업
- 를 사용하여 DynamoDB 작업 AWS SDK for Kotlin
  - AWS 계정 기반 엔드포인트 사용
  - DynamoDB Mapper(개발자 미리 보기)를 사용하여 클래스를 DynamoDB 항목에 매핑
    - DynamoDB Mapper 시작하기
      - 종속성 추가
      - 매퍼 생성 및 사용
      - 클래스 주석을 사용하여 스키마 정의
      - 작업 호출
        - 페이지가 매겨진 응답 작업
    - DynamoDB 매퍼 구성
      - 인터셉터 사용
        - 요청 파이프라인 이해
        - 후크
          - 읽기 전용 후크
          - 후크 수정

        - 실행 순서
        - 구성의 예
- 주석에서 스키마 생성
    - 플러그인 적용
    - 플러그인 구성
    - 클래스 주석 달기
        - 클래스 주석
        - 속성 주석
    - 사용자 지정 항목 변환기 정의
- 스키마 수동 정의
    - 코드에서 스키마 정의
- DynamoDB Mapper에서 보조 인덱스 사용
    - 보조 인덱스에 대한 스키마 정의
    - 작업에 보조 인덱스 사용
- 표현식 사용
    - 작업에 표현식 사용
    - DSL 구성 요소
        - 속성
        - 평등 및 불평등
        - 범위 및 세트
        - 부울 로직
        - 함수 및 속성
        - 정렬 키 필터

를 사용하여 Amazon S3 작업 AWS SDK for Kotlin

Kotlin용 Amazon Simple Storage Service의 기본 인터페이스SDK는 S3Client입니다. 의 다른 서비스 클라이언트와 `S3Client` 같은 SDK를 사용하여 Amazon S3에 요청합니다.

S3와 SDK 함께 Kotlin을 사용하는 데 도움이 되는 리소스는 다음과 같습니다.

- S3에 대한 Kotlin SDK 참조입니다. API S3

- S3 서비스 사용 설명서 및 서비스 API 참조.

다음 주제에서는 S3로 작동하는 일부 Kotlin에 대한 가이드 코드 예제SDKAPIs를 제공합니다.

주제

- 체크섬을 통한 데이터 무결성 보호
- KotlinSDK용를 사용하여 Amazon S3 다중 리전 액세스 포인트 작업

체크섬을 통한 데이터 무결성 보호

Amazon Simple Storage Service(S3)는 객체를 업로드할 때 체크섬을 지정하는 기능을 제공합니다. 체크섬을 지정하면 객체와 함께 저장되며 객체를 다운로드할 때 유효성을 검사할 수 있습니다.

체크섬은 파일을 전송할 때 데이터 무결성을 한층 더 강화합니다. 체크섬을 사용하면 수신된 파일이 원본 파일과 일치하는지 확인하여 데이터 일관성을 확인할 수 있습니다. Amazon S3를 사용하는 체크섬에 대한 자세한 내용은 지원되는 알고리즘을 포함한 Amazon Simple Storage Service 사용 설명서를 참조하세요. https://docs.aws.amazon.com/AmazonS3/latest/userguide/checking-object-integrity.html#using-additional-checksums

필요에 가장 적합한 알고리즘을 유연하게 선택하고 SDK가 체크섬을 계산하도록 할 수 있습니다. 또는 지원되는 알고리즘 중 하나를 사용하여 사전 계산된 체크섬 값을 제공할 수 있습니다.

> **Note**
>
> 의 버전 1.4.0부터 AWS SDK for Kotlin SDK는 업로드에 대한 CRC32 체크섬을 자동으로 계산하여 기본 무결성 보호를 제공합니다. SDK는 미리 계산된 체크섬 값을 제공하지 않거나 SDK가 체크섬을 계산하는 데 사용해야 하는 알고리즘을 지정하지 않은 경우이 체크섬을 계산합니다.
> 또한 SDK는 외부에서 설정할 수 있는 데이터 무결성 보호에 대한 전역 설정을 제공하며, 이는 AWS SDKs 및 도구 참조 안내서에서 확인할 수 있습니다.

체크섬은 객체 업로드와 객체 다운로드라는 두 가지 요청 단계로 설명합니다.

객체 업로드

요청 파라미터가 있는 putObject 함수를 사용하여 Kotlin용 SDK를 사용하여 Amazon S3에 객체를 업로드합니다. 요청 데이터 유형은 체크섬 계산을 가능하게 하는 checksumAlgorithm 속성을 제공합니다.

다음 코드 조각은 CRC32 체크섬이 있는 객체를 업로드하라는 요청을 보여줍니다. SDK가 요청을 보내면 CRC32 체크섬을 계산하고 객체를 업로드합니다. Amazon S3은 객체와 함께 체크섬을 저장합니다.

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

요청에 체크섬 알고리즘을 제공하지 않으면 체크섬 동작은 다음 표와 같이 사용하는 SDK 버전에 따라 달라집니다.

체크섬 알고리즘이 제공되지 않은 경우 체크섬 동작

Kotlin SDK 버전	체크섬 동작
1.4.0 이전	SDK는 CRC 기반 체크섬을 자동으로 계산하여 요청에 제공하지 않습니다.
1.4.0 이상	SDK는 CRC32 알고리즘을 사용하여 체크섬을 계산하고 요청에 제공합니다. Amazon S3는 자체 CRC32 체크섬을 계산하여 전송의 무결성을 검증하고 이를 SDK에서 제공하는 체크섬과 비교합니다. 체크섬이 일치하면 체크섬이 객체와 함께 저장됩니다.

미리 계산된 체크섬 값 사용

요청과 함께 제공되는 사전 계산된 체크섬 값은 SDK의 자동 계산을 비활성화하고 제공된 값을 대신 사용합니다.

다음 예제에서는 미리 계산된 SHA256 체크섬이 있는 요청을 보여줍니다.

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

Amazon S3에서 체크섬 값이 지정된 알고리즘에 대해 올바르지 않다고 판단하면 서비스는 오류 응답을 반환합니다.

멀티파트 업로드

멀티파트 업로드에 체크섬을 사용할 수도 있습니다.

CreateMultipartUpload 요청 및 각 UploadPart 요청에서 체크섬 알고리즘을 지정해야 합니다. 마지막 단계로 각 부분의 체크섬을 CompleteMultipartUpload에서 지정해야 합니다. 다음 예시는 지정된 체크섬 알고리즘을 사용하여 멀티파트 업로드를 만드는 방법을 보여줍니다.

```
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
    }
}
```

객체 다운로드

getObject getObject메서드를 사용하여 객체를 다운로드하면 에 대한 빌더의 checksumMode 속성GetObjectRequest이 로 설정된 경우 ChecksumMode.Enabled.

다음 스니펫의 요청은 체크섬을 계산하고 값을 비교하여 응답의 체크섬을 검증하도록 SDK에 지시합니다.

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

체크섬과 함께 객체를 업로드하지 않은 경우 검증이 수행되지 않습니다.

SDK 버전 1.4.0 이상을 사용하는 경우 SDK는 getObject 요청에 추가하지 않고 checksumMode = ChecksumMode.Enabled 요청의 무결성을 자동으로 확인합니다.

비동기식 검증

Kotlin용 SDK는 Amazon S3에서 객체를 다운로드할 때 스트리밍 응답을 사용하기 때문에 객체를 소비할 때 체크섬이 계산됩니다. 따라서 체크섬이 검증되도록 객체를 소비해야 합니다.

다음 예시는 응답을 완전히 소비하여 체크섬을 검증하는 방법을 보여줍니다.

```
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.
}
```

반면, 다음 예제의 코드에서는 객체를 전혀 사용하지 않으므로 체크섬이 검증되지 않습니다.

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

SDK에서 계산한 체크섬이 응답과 함께 전송된 예상 체크섬과 일치하지 않는 경우 SDK는 `ChecksumMismatchException`를 발생시킵니다.

KotlinSDK용를 사용하여 Amazon S3 다중 리전 액세스 포인트 작업

Amazon S3 다중 리전 액세스 포인트는 애플리케이션이 여러 AWS 리전리전에 있는 Amazon S3 버킷의 요청을 이행하는 데 사용할 수 있는 글로벌 엔드포인트를 제공합니다. 다중 리전 액세스 포인트를 사용하여 단일 리전에서 사용되는 것과 동일한 아키텍처로 다중 리전 애플리케이션을 구축하면 전 세계 어디에서나 해당 애플리케이션을 실행할 수 있습니다.

Amazon S3 사용 설명서에는 다중 리전 액세스 포인트에 대한 자세한 배경 정보가 포함되어 있습니다.

다중 리전 액세스 포인트 작업

다중 리전 액세스 포인트를 생성하려면 먼저 요청을 처리하려는 각 AWS 리전에 버킷을 하나씩 지정합니다. 다음 코드 조각은 버킷 2개를 생성합니다.

버킷 생성

다음 함수는 다중 리전 액세스 포인트와 함께 사용할 버킷 2개를 생성합니다. 한 버킷은 리전에 `us-east-1` 있고 다른 버킷은 리전에 있습니다`us-west-1`.

첫 번째 인수로 전달된 S3 클라이언트의 생성은 아래의 첫 번째 예제에 나와 있습니다the section called “ 객체 작업”.

```
        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.")
            }
        }
```

KotlinSDK의 S3 제어 클라이언트를 사용하여 다중 리전 액세스 포인트를 생성, 삭제 및 가져옵니다.

다음 코드 조각과 같이 S3 제어 아티팩트에 대한 종속성을 추가합니다. (*X.Y.Z*링크로 이동하여 사용 가능한 최신 버전을 볼 수 있습니다.)

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

```
implementation("aws.sdk.kotlin:s3control")
...
```

다음 코드와 같이 S3 제어 클라이언트를와 함께 AWS 리전 us-west-2 작동하도록 구성합니다. 모든 S3 제어 클라이언트 작업은 us-west-2 리전을 대상으로 해야 합니다.

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

S3 제어 클라이언트를 사용하여 다음 코드와 같이 버킷 이름(이전에 생성됨)을 지정하여 다중 리전 액세스 포인트를 생성합니다.

```
    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"
    }
```

다중 리전 액세스 포인트 생성은 비동기 작업이므로 즉각적인 응답에서 받은 토큰을 사용하여 생성 프로세스의 상태를 확인합니다. 상태 확인에서 성공 메시지가 반환되면 `GetMultiRegionAccessPoint` 작업을 사용하여 다중 리전 액세스 포인트의 별칭을 가져올 수 있습니다. 별칭은 객체 수준 작업에 ARN필요한의 마지막 구성 요소입니다.

토큰을 사용하여 상태 확인

`DescribeMultiRegionAccessPointOperation`를 사용하여 마지막 작업의 상태를 확인합니다. `requestStatus` 값이 “SUCCEEDED”가 되면 다중 리전 액세스 포인트로 작업할 수 있습니다.

```
        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)
            }
        }
```

객체 및 다중 리전 액세스 포인트 작업

S3 클라이언트를 사용하여 다중 리전 액세스 포인트의 객체로 작업합니다. 다중 리전 액세스 포인트에서 사용할 수 있는 버킷의 객체에 사용하는 많은 작업. 자세한 내용과 전체 작업 목록은 S3 작업과의 다중 리전 액세스 포인트 호환성을 참조하세요.

다중 리전 액세스 포인트를 사용하는 작업은 비대칭 Sigv4(Sigv4a) 서명 알고리즘으로 서명됩니다. 에서 Sigv4a를 지원하려면 AWS SDK for Kotlin 현재 별도의 종속성인 CRT 서명자로 서명해야 합니다. Sigv4a에 대한 지원을 구성하려면 프로젝트에 다음 종속성을 추가합니다. (*X.Y.Z*링크로 이동하여 사용 가능한 최신 버전을 볼 수 있습니다.)

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

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

종속성을 추가한 후 다음 코드와 같이 Sigv4a 서명 알고리즘을 사용하도록 SS3 클라이언트를 구성합니다.

```
        suspend fun createS3Client(): S3Client {
            // Configure your S3Client to use the Asymmetric Sigv4 (Sigv4a) signing
 algorithm.
            val sigV4AScheme = SigV4AsymmetricAuthScheme(CrtAwsSigner)
```

```
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4AScheme)
            }
            return s3
        }
```

S3 클라이언트를 구성한 후 S3가 다중 리전 액세스 포인트에 대해 지원하는 작업은 동일하게 작동합니다. 유일한 차이점은 버킷 파라미터가 다중 리전 액세스 포인트ARN의 이어야 한다는 것입니다. 를 반환하는 `createMrap` 함수의 앞부분과 같이 Amazon S3 콘솔ARN에서 또는 프로그래밍 방식으로를 가져올 수 있습니다ARN.

다음 코드 예제는 `GetObject` 작업에 ARN 사용되는를 보여줍니다.

```
    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
    }
```

를 사용하여 DynamoDB 작업 AWS SDK for Kotlin

AWS 계정 기반 엔드포인트 사용

DynamoDB는 AWS 계정 ID를 사용하여 요청 라우팅을 간소화하여 성능을 개선할 수 있는 계정 기반 엔드포인트를 제공합니다. AWS

이 기능을 활용하려면 버전 1.3.37 이상의를 사용해야 합니다 AWS SDK for Kotlin. Maven 중앙 리포지토리에 SDK 나열된의 최신 버전을 찾을 수 있습니다. 지원되는 버전의 SDK가 활성화되면 자동으로 새 엔드포인트를 사용합니다.

계정 기반 라우팅을 옵트아웃하려면 다음 네 가지 옵션이 있습니다.

- 가 로 `AccountIdEndpointMode` 설정된 DynamoDB 서비스 클라이언트를 구성합니다`DISABLED`.
- 환경 변수를 설정합니다.
- JVM 시스템 속성을 설정합니다.
- 공유 AWS 구성 파일 설정을 업데이트합니다.

다음 코드 조각은 DynamoDB 서비스 클라이언트를 구성하여 계정 기반 라우팅을 비활성화하는 방법의 예입니다.

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

및 도구 참조 가이드는 AWS SDKs 마지막 세 가지 구성 옵션에 대한 자세한 정보를 제공합니다.

DynamoDB Mapper(개발자 미리 보기)를 사용하여 클래스를 DynamoDB 항목에 매핑

⚠ DynamoDB Mapper는 개발자 미리 보기 릴리스입니다. 기능이 완전하지 않으며 변경될 수 있습니다.

DynamoDB Mapper는 AWS SDK for Java의 DynamoDB Enhanced Client 또는의 객체 지속성 모델과 유사하게 Kotlin 클래스를 DynamoDB 테이블 및 인덱스에 매핑하는 메커니즘 AWS SDK for .NET을 제공하는 상위 수준 라이브러리입니다. https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DotNetSDKHighLevel.html

데이터 객체와 이를 DynamoDB 항목으로 변환하는 방법을 설명하는 스키마를 정의합니다. 스키마를 정의한 후 DynamoDB Mapper는 테이블 및 인덱스에서 객체를 생성, 읽기, 업데이트 또는 삭제(CRUD) 작업에 사용할 수 있는 직관적인 인터페이스를 제공합니다.

주제

- DynamoDB Mapper 시작하기
- DynamoDB 매퍼 구성
- 주석에서 스키마 생성
- 스키마 수동 정의
- DynamoDB Mapper에서 보조 인덱스 사용
- 표현식 사용

DynamoDB Mapper 시작하기

⚠ DynamoDB Mapper는 개발자 미리 보기 릴리스입니다. 기능이 완전하지 않으며 변경될 수 있습니다.

다음 자습서에서는 DynamoDB Mapper의 기본 구성 요소를 소개하고 코드에 사용하는 방법을 보여줍니다.

종속성 추가

Gradle 프로젝트에서 DynamoDB Mapper 작업을 시작하려면 `build.gradle.kts` 파일에 플러그인과 두 개의 종속성을 추가합니다.

(*X.Y.Z* 링크로 이동하여 사용 가능한 최신 버전을 볼 수 있습니다.)

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

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

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

*<Version>을 최신 SDK 릴리스로 바꿉니다. SDK의 최신 버전을 찾으려면 GitHub에서 최신 릴리스를 확인하세요.

> **Note**
>
> 스키마를 수동으로 정의하려는 경우 이러한 종속성 중 일부는 선택 사항입니다. the section called “스키마 수동 정의” 자세한 내용과 감소된 종속성 세트는 섹션을 참조하세요.

매퍼 생성 및 사용

DynamoDB Mapper는 AWS SDK for Kotlin의 DynamoDB 클라이언트를 사용하여 DynamoDB와 상호 작용합니다. 다음 코드 조각과 같이 매퍼 `DynamoDbClient` 인스턴스를 생성할 때 완전히 구성된 인스턴스를 제공해야 합니다.

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

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

매퍼 인스턴스를 생성한 후에는 다음과 같이 매퍼 인스턴스를 사용하여 테이블 인스턴스를 가져올 수 있습니다.

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

이전 코드는에서 정의한 스키마가 `cars` 있는 DynamoDB의 테이블에 대한 참조를 가져옵니다`CarSchema`(아래에서 스키마에 대해 설명). 테이블 인스턴스를 생성한 후 이에 대한 작업을 수행할 수 있습니다. 다음 코드 조각은 `cars` 테이블에 대한 두 가지 예제 작업을 보여줍니다.

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

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

이전 코드는 cars 테이블에 새 항목을 생성합니다. 코드는 Car 클래스를 사용하여 Car 인스턴스를 인라인으로 생성하며, 그 정의는 아래에 나와 있습니다. 그런 다음 코드는 파티션 키가 인 항목을 cars 테이블에서 쿼리Peugeot하고 인쇄합니다. 작업은 아래에 자세히 설명되어 있습니다.

클래스 주석을 사용하여 스키마 정의

다양한 Kotlin 클래스의 경우 SDK는 Gradle용 DynamoDB Mapper Schema Generator 플러그인을 사용하여 빌드 시 스키마를 자동으로 생성할 수 있습니다. 스키마 생성기를 사용하면 SDK가 클래스를 검사하여 스키마를 추론하여 스키마를 수동으로 정의하는 데 수반되는 일부 보일러플레이트가 완화됩니다. 추가 주석 및 구성을 사용하여 생성된 스키마를 사용자 지정할 수 있습니다. ???

주석에서 스키마를 생성하려면 먼저 클래스에 및 @DynamoDbItem 키를 사용하여 주석을 달@DynamoDbPartitionKey고 키에 및를 사용합니다@DynamoDbSortKey. 다음 코드는 주석이 달린 Car 클래스를 보여줍니다.

```
// 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
)
```

빌드 후 자동으로 생성된를 참조할 수 있습니다CarSchema. 매퍼의 getTable 메서드에 있는 참조를 사용하여 다음과 같이 테이블 인스턴스를 가져올 수 있습니다.

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

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

또는 빌드 시 DynamoDbMapper 자동으로 생성되는의 확장 방법을 활용하여 테이블 인스턴스를 가져올 수 있습니다. 이 접근 방식을 사용하면 스키마를 이름으로 참조할 필요가 없습니다. 다음과 같이 자동으로 생성된 getCarsTable 확장 메서드는 테이블 인스턴스에 대한 참조를 반환합니다.

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

자세한 내용과 예는 the section called “스키마 생성” 단원을 참조하십시오.

작업 호출

DynamoDB Mapper는 SDK의에서 사용할 수 있는 작업의 하위 집합을 지원합니다DynamoDbClient. 매퍼 작업의 이름은 SDK 클라이언트의 해당 작업과 동일합니다. 많은 매퍼 요청/응답 멤버는 SDK 클라이언트 멤버와 동일하지만 일부 멤버는 이름 변경, 다시 입력 또는 완전히 삭제되었습니다.

다음과 같이 DSL 구문을 사용하여 테이블 인스턴스에서 작업을 호출합니다.

```
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)
```

명시적 요청 객체를 사용하여 작업을 호출할 수도 있습니다.

```
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)
```

이전 두 코드 예제는 동일합니다.

페이지가 매겨진 응답 작업

query 및와 같은 일부 작업은 너무 커서 단일 응답으로 반환할 수 없는 데이터 컬렉션을 반환할 scan 수 있습니다. 모든 객체가 처리되도록 DynamoDB Mapper는 DynamoDB를 즉시 호출하지 않고 대신 다음과 같은 작업 응답 유형의 Flow를 반환하는 페이지 매김 메Flow<ScanResponse<Car>>서드를 제공합니다.

```
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) }
}
```

종종 객체 흐름은 객체를 포함하는 응답 흐름보다 비즈니스 로직에 더 유용합니다. 매퍼는 페이지가 매겨진 응답에 대한 확장 방법을 제공하여 객체의 흐름에 액세스합니다. 예를 들어 다음 코드는 앞에서 설명한 `Flow<ScanResponse<Car>>` 대로 `Flow<Car>` 대신를 반환합니다.

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

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

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

DynamoDB 매퍼 구성

> ⚠ DynamoDB Mapper는 개발자 미리 보기 릴리스입니다. 기능이 완전하지 않으며 변경될 수 있습니다.

DynamoDB Mapper는 애플리케이션에 맞게 라이브러리의 동작을 사용자 지정할 수 있는 구성 옵션을 제공합니다.

인터셉터 사용

DynamoDB Mapper 라이브러리는 매퍼 요청 파이프라인의 중요한 단계에서 활용할 수 있는 후크를 정의합니다. `Interceptor` 인터페이스를 구현하여 매퍼 프로세스를 관찰하거나 수정하는 후크를 구현할 수 있습니다.

단일 DynamoDB Mapper에 하나 이상의 인터셉터를 구성 옵션으로 등록할 수 있습니다. 인터셉터를 등록하는 방법은이 섹션의 끝에 있는 예제를 참조하세요.

요청 파이프라인 이해

매퍼의 요청 파이프라인은 다음 5단계로 구성됩니다.

1. 초기화: 작업을 설정하고 초기 컨텍스트를 수집합니다.
2. 직렬화: 상위 수준 요청 객체를 하위 수준 요청 객체로 변환합니다. 이 단계에서는 상위 수준 Kotlin 객체를 속성 이름 및 값으로 구성된 DynamoDB 항목으로 변환합니다.
3. 하위 수준 호출: 기본 DynamoDB 클라이언트에서 요청을 실행합니다.
4. 역직렬화: 하위 수준 응답 객체를 상위 수준 응답 객체로 변환합니다. 이 단계에는 속성 이름 및 값으로 구성된 DynamoDB 항목을 상위 수준 Kotlin 객체로 변환하는 작업이 포함됩니다.
5. 완료: 호출자에게 반환할 상위 수준 응답을 완료합니다. 파이프라인 실행 중에 예외가 발생하면이 단계에서는 호출자에게 발생하는 예외를 완료합니다.

후크

후크는 매퍼가 파이프라인의 특정 단계 전후에 호출하는 인터셉터 메서드입니다. 후크에는 읽기 전용과 수정(또는 읽기-쓰기)의 두 가지 변형이 있습니다. 예를 들어, `readBeforeInvocation`는 낮은 수준의 호출 단계 전에 매퍼가 단계에서 실행하는 읽기 전용 후크입니다.

읽기 전용 후크

매퍼는 파이프라인의 각 단계 전후에 읽기 전용 후크를 호출합니다(초기화 단계 이전 및 완료 단계 이후 제외). 읽기 전용 후드는 진행 중인 상위 수준 작업에 대한 읽기 전용 보기를 제공합니다. 예를 들어 로깅, 디버깅, 지표 수집을 위해 작업의 상태를 검사하는 메커니즘을 제공합니다. 각 읽기 전용 후크는 컨텍스트 인수를 수신하고를 반환합니다`Unit`.

매퍼는 읽기 전용 후크 중에 발생하는 모든 예외를 포착하여 컨텍스트에 추가합니다. 그런 다음 동일한 단계의 후속 인터셉터 후크에 예외를 제외하고 컨텍스트를 전달합니다. 매퍼는 동일한 단계에 대해 마지막 인터셉터의 읽기 전용 후크를 호출한 후에만 호출자에게 예외를 발생시킵니다. 예를 들어 매퍼가 인터셉터 A 2개와 로 구성B되고 A의 `readAfterSerialization` 후크가 예외를 발생시키는 경우 매퍼는 B의 `readAfterSerialization` 후크에 전달된 컨텍스트에 예외를 추가합니다. B의 `readAfterSerialization` 후크가 완료되면 매퍼는 호출자에게 예외를 다시 발생시킵니다.

후크 수정

매퍼는 파이프라인의 각 단계 전에 수정 후크를 호출합니다(초기화 전 제외). 후크 수정은 진행 중인 상위 수준 작업을 보고 수정할 수 있는 기능을 제공합니다. 이는 매퍼 구성 및 항목 스키마가 제공하지 않는 방식으로 동작 및 데이터를 사용자 지정하는 데 사용할 수 있습니다. 각 수정 후크는 컨텍스트 인수를 수신하고 그 결과 해당 컨텍스트의 일부 하위 집합을 반환합니다. 이는 후크에 의해 수정되거나 입력 컨텍스트에서 전달된 것입니다.

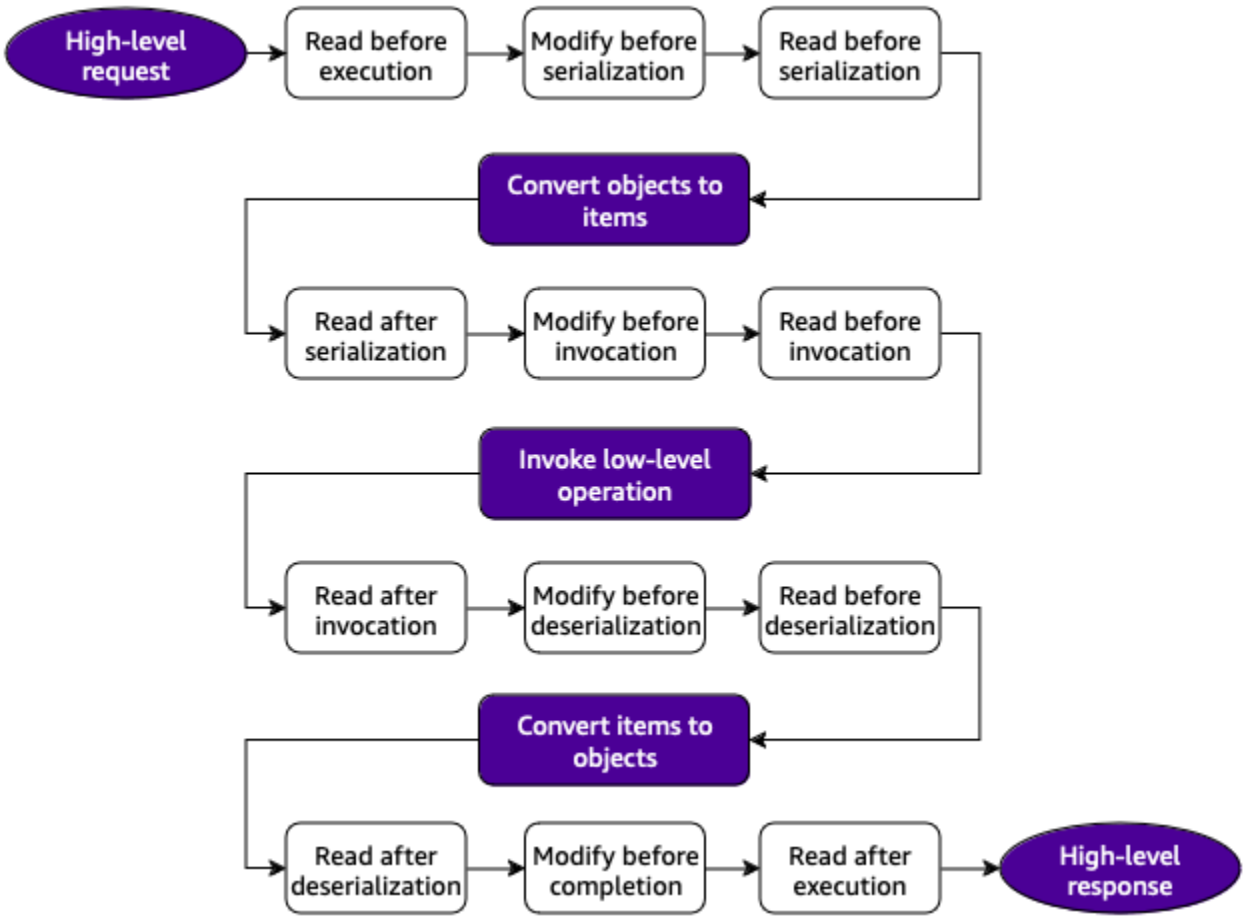
수정 후크를 실행하는 동안 매퍼가 예외를 포착하면 동일한 단계에서 다른 인터셉터의 수정 후크를 실행하지 않습니다. 매퍼는 컨텍스트에 예외를 추가하고 다음 읽기 전용 후크로 전달합니다. 매퍼는 동일한 단계에 대해 마지막 인터셉터의 읽기 전용 후크를 호출한 후에만 호출자에게 예외를 발생시킵니다. 예를 들어 매퍼가 인터셉터 A 2개와 로 구성B되고 A의 `modifyBeforeSerialization` 후크가 예외를 발생시키는 경우 B의 `modifyBeforeSerialization` 후크는 호출되지 않습니다. 인터셉터 A및 후크B'`readAfterSerialization`가 실행되며, 그 후에는 호출자에게 예외가 다시 발생합니다.

실행 순서

인터셉터가 매퍼의 구성에 정의된 순서에 따라 매퍼가 후크를 호출하는 순서가 결정됩니다.

- 하위 수준 호출 단계 이전의 단계의 경우 구성에 추가된 것과 동일한 순서로 후크를 실행합니다.
- 하위 수준 호출 단계 이후의 단계의 경우 구성에 추가된 순서와 역순으로 후크를 실행합니다.

다음 다이어그램은 후크 메서드의 실행 순서를 보여줍니다.



후크 메서드의 실행 순서에 대한 텍스트 설명

매퍼는 인터셉터의 후크를 다음 순서로 실행합니다.

1. DynamoDB Mapper가 상위 수준 요청을 호출합니다.
2. 실행 전 읽기
3. 직렬화 전 수정
4. 직렬화 전 읽기
5. DynamoDB Mapper가 객체를 항목으로 변환
6. 직렬화 후 읽기
7. 호출 전 수정
8. 호출 전 읽기
9. DynamoDB Mapper는 하위 수준 작업을 호출합니다.
10. 호출 후 읽기
11. 역직렬화 전 수정

12.역직렬화 전 읽기

13DynamoDB Mapper는 항목을 객체로 변환합니다.

14.역직렬화 후 읽기

15.완료 전 수정

16.실행 후 읽기

17DynamoDB Mapper가 상위 수준 응답을 반환합니다.

구성의 예

다음 예제에서는 `DynamoDbMapper` 인스턴스에서 인터셉터를 구성하는 방법을 보여줍니다.

```
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
}
```

주석에서 스키마 생성

⚠ DynamoDB Mapper는 개발자 미리 보기 릴리스입니다. 기능이 완전하지 않으며 변경될 수 있습니다.

DynamoDB Mapper는 Kotlin 클래스와 DynamoDB 항목 간의 매핑을 정의하는 스키마를 사용합니다. Kotlin 클래스는 스키마 생성기 Gradle 플러그인을 사용하여 스키마 생성을 주도할 수 있습니다.

플러그인 적용

클래스에 대한 코드 생성 스키마를 시작하려면 애플리케이션의 빌드 스크립트에 플러그인을 적용하고 주석 모듈에 종속성을 추가합니다. 다음 Gradle 스크립트 조각은 코드 생성에 필요한 설정을 보여줍니다.

(*X.Y.Z*링크로 이동하여 사용 가능한 최신 버전을 볼 수 있습니다.)

```
// 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")
}
```

플러그인 구성

플러그인은 빌드 스크립트에서 `dynamoDbMapper { ... }` 플러그인 확장을 사용하여 적용할 수 있는 다양한 구성 옵션을 제공합니다.

옵션	옵션 설명	값
`generateBuilderClasses`	주석이 달린 클래스에 대해 DSL스타일 빌더 클래스를 생성할지 여부를 제어합니다. `@DynamoDbItem`	`WHEN_REQUIRED` (기본값): 퍼블릭 변경 가능 멤버로만 구성되고 제로 구성 생성자가 있는 클래스에는 빌더 클래스가 생성되지 않습니다. `ALWAYS`: Builder 클래스가 항상 생성됩니다.

옵션	옵션 설명	값
visibility	생성된 클래스의 가시성을 제어합니다.	PUBLIC(기본값) INTERNAL
destinationPackage	생성된 클래스의 패키지 이름을 지정합니다.	RELATIVE (기본값): 스키마 클래스는 주석이 달린 클래스를 기준으로 하위 패키지로 생성됩니다. 기본적으로 하위 패키지의 이름은 dynamodbmapper.generatedschemas 이며 문자열 파라미터를 전달하여 구성할 수 있습니다. ABSOLUTE: 스키마 클래스는 애플리케이션의 루트를 기준으로 절대 패키지로 생성됩니다. 기본적으로 패키지의 이름은 이며 aws.sdk.kotlin.hll.dynamodbmapper.generatedschemas 문자열 파라미터를 전달하여 구성할 수 있습니다.
generateGetTableExtension	DynamoDbMapper.get${CLASS_NAME}Table 확장 메서드 생성 여부 제어	true(기본값) false

Example 코드 생성 플러그인 구성의 예

다음 예제에서는 생성된 스키마의 대상 패키지와 가시성을 구성합니다.

```
// 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
}
```

클래스 주석 달기

스키마 생성기는 클래스 주석을 찾아 스키마를 생성할 클래스를 결정합니다. 스키마 생성을 옵트인하려면를 사용하여 클래스에 주석을 지정합니다@DynamoDbItem. 또한 항목의 파티션 키 역할을 하는 클래스 속성에 주석을 달아야 합니다@DynamoDbPartitionKey.

다음 클래스 정의는 스키마 생성에 필요한 최소한의 주석을 보여줍니다.

Example

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

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

클래스 주석

스키마 생성을 제어하기 위해 클래스에 다음 주석이 적용됩니다.

- @DynamoDbItem:이 클래스/인터페이스가 테이블의 항목 유형을 설명하도록 지정합니다. 이 유형의 모든 퍼블릭 속성은 명시적으로 무시되지 않는 한 속성에 매핑됩니다. 이 클래스가 있으면이 클래스에 대한 스키마가 생성됩니다.
  - converterName: 스키마 생성기 플러그인에서 생성한 스키마 대신 사용자 지정 스키마를 사용해야 함을 나타내는 선택적 파라미터입니다. 사용자 지정 ItemConverter 클래스의 정규화된 이름입니다. 이 the section called “사용자 지정 항목 변환기 정의” 섹션에서는 사용자 지정 스키마를 생성하고 사용하는 예를 보여줍니다.

속성 주석

클래스 속성에 다음 주석을 적용하여 스키마 생성을 제어할 수 있습니다.

- @DynamoDbPartitionKey: 항목의 파티션 키를 지정합니다.
- @DynamoDbSortKey: 항목에 대한 선택적 정렬 키를 지정합니다.
- @DynamoDbIgnore: DynamoDB Mapper에서이 클래스 속성을 항목 속성으로 변환하거나 항목 속성에서 변환하지 않도록 지정합니다.
- @DynamoDbAttribute:이 클래스 속성에 대한 선택적 사용자 지정 속성 이름을 지정합니다.

사용자 지정 항목 변환기 정의

경우에 따라 클래스에 대한 사용자 지정 항목 변환기를 정의할 수 있습니다. 이는 클래스가 스키마 생성기 플러그인에서 지원하지 않는 유형을 사용하는 경우 한 가지 이유입니다. 다음 버전의 `Employee` 클래스를 예로 사용합니다.

```
import kotlin.uuid.Uuid

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

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

이제 `Employee` 클래스는 스키마 생성기에서 현재 지원되지 않는 `kotlin.uuid.Uuid` 유형을 사용합니다. 스키마 생성이 실패하고 오류가 발생합니다`Unsupported attribute type TypeRef(pkg=kotlin.uuid, shortName=Uuid, genericArgs=[], nullable=false)`. 이 오류는 플러그인이이 클래스에 대한 항목 변환기를 생성할 수 없음을 나타냅니다. 따라서 자체적으로 작성해야 합니다.

이를 위해 클래스에 `ItemConverter` 대한를 구현한 다음 새 항목 변환기의 정규화된 이름을 지정하여 `@DynamoDbItem` 클래스 주석을 수정합니다.

먼저 `kotlin.uuid.Uuid` 클래스에 `ValueConverter` 대해를 구현합니다.

```
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())
}
```

그런 다음 Employee 클래스ItemConverter에 대해를 구현합니다. 는 "workstationId"에 대한 속성 설명자에서이 새 값 변환기를 ItemConverter 사용합니다.

```
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
        )
    ),
)
```

이제 항목 변환기를 정의했으므로 클래스에 적용할 수 있습니다. 다음과 같이 정규화된 클래스 이름을 제공하여 항목 변환기를 참조하도록 @DynamoDbItem 주석을 업데이트합니다.

```
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
)
```

마지막으로 DynamoDB Mapper에서 클래스 사용을 시작할 수 있습니다.

스키마 수동 정의

⚠ DynamoDB Mapper는 개발자 미리 보기 릴리스입니다. 기능이 완전하지 않으며 변경될 수 있습니다.

코드에서 스키마 정의

제어 및 사용자 지정 가능성을 극대화하기 위해 코드에서 스키마를 수동으로 정의하고 사용자 지정할 수 있습니다.

다음 코드 조각과 같이 주석 기반 스키마 생성을 사용할 때보다 `build.gradle.kts` 파일에 더 적은 종속성을 포함해야 합니다.

(*X.Y.Z* 링크로 이동하여 사용 가능한 최신 버전을 볼 수 있습니다.)

```
// 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.
}
```

스키마 생성기 플러그인이나 주석 패키지는 필요하지 않습니다.

Kotlin 클래스와 DynamoDB 항목 간의 매핑에는 `ItemSchema<T>` 구현이 필요합니다. 여기서 `T`는 Kotlin 클래스의 유형입니다. 스키마는 다음 요소로 구성됩니다.

- Kotlin 객체 인스턴스와 DynamoDB 항목 간에 변환하는 방법을 정의하는 항목 변환기입니다.
- 파티션 키 속성의 이름과 유형을 정의하는 파티션 키 사양입니다.
- 선택적으로 정렬 키 사양으로, 정렬 키 속성의 이름과 유형을 정의합니다.

다음 코드에서는 `CarSchema` 인스턴스를 수동으로 생성합니다.

```
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)
```

이전 코드는 라는 변환기를 생성합니다. `carConverter`이 변환기는의 익명 구현으로 정의됩니다`ItemConverter<Car>`. 변환기의 `convertTo` 메서드는 `Car` 인수를 수락하고 DynamoDB 항목 속성의 리터럴 키와 값을 나타내는 `Item` 인스턴스를 반환합니다. 변환기의 `convertFrom` 메서드는 `Item` 인수를 수락하고 `Item` 인수의 속성 값에서 `Car` 인스턴스를 반환합니다.

다음으로 코드는 파티션 키와 정렬 키의 두 가지 키 사양을 생성합니다. 모든 DynamoDB 테이블 또는 인덱스에는 정확히 하나의 파티션 키가 있어야 하며, 이에 따라 모든 DynamoDB Mapper 스키마 정의가 있어야 합니다. 스키마에는 정렬 키가 하나 있을 수도 있습니다.

마지막 문에서 코드는 변환기 및 키 사양에서 `cars` DynamoDB 테이블에 대한 스키마를 생성합니다.

결과 스키마는 the section called “클래스 주석을 사용하여 스키마 정의” 섹션에서 생성한 주석 기반 스키마와 동일합니다. 참고로 다음은 사용된 주석이 달린 클래스입니다.

DynamoDB Mapper 주석이 있는 자동차 클래스

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

    @DynamoDbSortKey
    val model: String,

    val initialYear: Int
)
```

`ItemConverter`DynamoDB Mapper에는 자체를 구현하는 것 외에도 다음과 같은 몇 가지 유용한 구현이 포함되어 있습니다.

- `SimpleItemConverter`:는 빌더 클래스 및 속성 설명자를 사용하여 간단한 변환 로직을 제공합니다. 이 구현을 사용하는 the section called “사용자 지정 항목 변환기 정의” 방법은의 예제를 참조하세요.
- `HeterogeneousItemConverter`:는 구분자 속성을 사용하고 하위 유형에 대한 `ItemConverter` 인스턴스를 위임하여 다형성 유형 변환 로직을 제공합니다.
- `DocumentConverter`: `Document` 객체의 비정형 데이터에 대한 변환 로직을 제공합니다.

DynamoDB Mapper에서 보조 인덱스 사용

⚠ DynamoDB Mapper는 개발자 미리 보기 릴리스입니다. 기능이 완전하지 않으며 변경될 수 있습니다.

보조 인덱스에 대한 스키마 정의

DynamoDB 테이블은 기본 테이블 자체에 정의된 것과 다른 키를 사용하여 데이터에 대한 액세스를 제공하는 보조 인덱스를 지원합니다. 기본 테이블과 마찬가지로 DynamoDB Mapper는 `ItemSchema` 유형을 사용하여 인덱스와 상호 작용합니다.

DynamoDB 보조 인덱스는 기본 테이블의 모든 속성을 포함할 필요가 없습니다. 따라서 인덱스에 매핑되는 Kotlin 클래스는 해당 인덱스의 기본 테이블에 매핑되는 Kotlin 클래스와 다를 수 있습니다. 이 경우 인덱스 클래스에 대해 별도의 스키마를 선언해야 합니다.

다음 코드는 DynamoDB `cars` 테이블에 대한 인덱스 스키마를 수동으로 생성합니다.

```
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
        )
*/
```

이제 작업에서 Model 인스턴스를 사용할 수 있습니다.

작업에 보조 인덱스 사용

DynamoDB Mapper는 인덱스, 즉 queryPaginated 및에 대한 작업의 하위 집합을 지원합니다scanPaginated. 인덱스에서 이러한 작업을 호출하려면 먼저 테이블 객체에서 인덱스에 대한 참조를 얻어야 합니다. 다음 샘플에서는 cars-by-model 인덱스에 대해 이전에 생성한를 사용합니다(생성modelSchema은 여기에 표시되지 않음).

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

val modelFlow = index
```

```
    .scanPaginated { }
    .items()

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

표현식 사용

⚠ DynamoDB Mapper는 개발자 미리 보기 릴리스입니다. 기능이 완전하지 않으며 변경될 수 있습니다.

특정 DynamoDB 작업은 제약 조건이나 조건을 지정하는 데 사용할 수 있는 표현식을 허용합니다. DynamoDB Mapper는 표현식을 생성할 DSL 수 있는 고유한 Kotlin을 제공합니다. DSL는 코드에 더 큰 구조와 가독성을 제공하며 표현식을 더 쉽게 작성할 수 있습니다.

이 섹션에서는 DSL 구문을 설명하고 다양한 예제를 제공합니다.

작업에 표현식 사용

와 같은 작업에서 표현식을 사용합니다. `scan`여기서 표현식은 정의한 기준에 따라 반환된 항목을 필터링합니다. DynamoDB Mapper에서 표현식을 사용하려면 작업 요청에 표현식 구성 요소를 추가합니다.

다음 코드 조각은 `scan` 작업에 사용되는 필터 표현식의 예를 보여줍니다. lambda 인수를 사용하여 `year` 속성 값이 2001인 항목으로 반환할 항목을 제한하는 필터 기준을 설명합니다.

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

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

다음 예제는 정렬 키 필터링과 비키 필터링이라는 두 위치에서 표현식을 지원하는 `query` 작업을 보여줍니다.

```
table.queryPaginated {
    keyCondition = KeyFilter(partitionKey = 1000) { sortKey startsWith "M" }
```

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

이전 코드는 세 가지 기준을 모두 충족하는 로 결과를 필터링합니다.

- 파티션 키 속성 값은 1000-AND-입니다.
- 정렬 키 속성 값은 문자 M-AND-로 시작합니다.
- 연도 속성 값은 2001입니다.

DSL 구성 요소

DSL 구문은 아래에 설명된 몇 가지 유형의 구성 요소를 표시하며 표현식을 빌드하는 데 사용합니다.

속성

대부분의 조건은 키 또는 문서 경로로 식별되는 속성을 참조합니다. 를 사용하면 `attr` 함수를 사용하여 모든 속성 참조를 DSK생성하고 선택적으로 추가 수정을 수행합니다.

다음 코드는 인덱스별 목록 디 참조 및 키별 맵 디 참조와 같이 간단한 속성 참조부터 복잡한 속성 참조까지 다양한 예제를 보여줍니다.

```
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`.
```

평등 및 불평등

균등성 및 비균등성별로 표현식의 속성 값을 비교할 수 있습니다. 속성 값을 리터럴 값 또는 기타 속성 값과 비교할 수 있습니다. 조건을 지정하는 데 사용하는 함수는 다음과 같습니다.

- eq:와 같음(과 동등==)
- neq:와 같지 않음(과 동등!=)
- gt: 보다 큽니다(과 동등>).

- gte:가 보다 크거나 같음(과 동등>=)
- lt: 미만(과 동등<)
- lte: 이하(과 동등<=)

다음 예제와 같이 인픽스 표기법을 사용하여 비교 함수를 인수와 결합합니다.

```
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`.
```

범위 및 세트

단일 값 외에도 속성 값을 범위 또는 세트의 여러 값과 비교할 수 있습니다. 다음 예제와 같이 infix isIn 함수를 사용하여 비교를 수행합니다.

```
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 함수는 수집(예: Set<String>) 및 Kotlin으로 표현할 수 있는 경계ClosedRange<T>(예:)에 대한 과부하를 제공합니다IntRange. 로 표현할 수 없는 경계ClosedRange<T>(예: 바이트 배열 또는 기타 속성 참조)의 경우 isBetween 함수를 사용할 수 있습니다.

```
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`.
```

부울 로직

다음 함수를 사용하여 부울 로직을 사용하여 개별 조건을 결합하거나 변경할 수 있습니다.

- and: 모든 조건이 true(같음 &&)여야 합니다.
- or: 하나 이상의 조건이 true여야 합니다(와 동일||).
- not: 지정된 조건은 false여야 합니다(와 동일!).

다음 예제는 각 함수를 보여줍니다.

```
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.
)
```

다음 표현식과 같이 부울 함수별로 부울 조건을 추가로 결합하여 중첩 로직을 생성할 수 있습니다.

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

```
)
```

이전 표현식은 다음 조건 중 하나를 충족하는 결과를 필터링합니다.

- 다음 두 조건이 모두 참입니다.
  - `foo` 속성 값은 123-AND-입니다.
  - `bar` 속성 값은 'abc'입니다.
- 다음 두 조건이 모두 참입니다.
  - `foo` 속성 값은 234-AND-입니다.
  - `bar` 속성 값은 'bcd'입니다.

이는 다음 Kotlin 부울 표현식과 동일합니다.

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

함수 및 속성

다음 함수 및 속성은 추가 표현식 기능을 제공합니다.

- `contains`: 문자열/목록 속성 값에 지정된 값이 포함되어 있는지 확인합니다.
- `exists`: 속성이 정의되어 있는지 확인하고 값을 보유합니다(포함`null`).
- `notExists`: 속성이 정의되지 않았는지 확인합니다.
- `isOfType`: 속성 값이 문자열, 숫자, 부울 등과 같은 지정된 유형인지 확인합니다.
- `size`: 모음의 요소 수 또는 문자열 길이와 같은 속성의 크기를 가져옵니다.
- `startsWith`: 문자열 속성 값이 지정된 하위 문자열로 시작하는지 확인합니다.

다음 예제에서는 표현식에 사용할 수 있는 추가 함수 및 속성의 사용을 보여줍니다.

```
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.
```

정렬 키 필터

정렬 키의 필터 표현식(예: query 작업의 keyCondition 파라미터)은 명명된 속성 값을 사용하지 않습니다. 필터에서 정렬 키를 사용하려면 모든 비교sortKey에서 키워드를 사용해야 합니다. 다음 예제와 attr("<sort key name>") 같이 sortKey 키워드가를 대체합니다.

```
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).
```

정렬 키 필터를 부울 로직과 결합할 수 없으며 위에서 설명한 비교의 하위 집합만 지원합니다.

- 평등 및 비형평성: 지원되는 모든 비교
- 범위 및 세트: 지원되는 모든 비교
- 부울 로직: 지원되지 않음
- 함수 및 속성: 만 지원startsWith됩니다.

SDK for Kotlin 코드 예제

이 주제의 코드 예제에서는 AWS SDK for Kotlin을 함께 사용하는 방법을 보여줍니다 AWS.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

일부 서비스에는 서비스와 관련된 라이브러리 또는 함수를 활용하는 방법을 보여주는 추가 예제 범주가 포함되어 있습니다.

서비스

- SDK for Kotlin을 사용한 API Gateway 예제
- SDK for Kotlin을 사용한 Aurora 예제
- SDK for Kotlin을 사용한 Auto Scaling 예제
- SDK for Kotlin를 사용한 Amazon Bedrock 예시
- SDK for Kotlin을 사용한 CloudWatch 예제
- SDK for Kotlin을 사용한 CloudWatch Logs 예제
- SDK for Kotlin을 사용한 Amazon Cognito 자격 증명 공급자용 코드 예제
- SDK for Kotlin을 사용한 Amazon Comprehend 예제
- SDK for Kotlin을 사용한 DynamoDB 예제
- SDK for Kotlin을 사용한 Amazon EC2 예제
- SDK for Kotlin을 사용한 Amazon ECR 예제
- SDK for Kotlin을 사용한 OpenSearch 서비스 예제
- SDK for Kotlin을 사용한 EventBridge 예제
- AWS Glue SDK for Kotlin을 사용한 예제
- SDK for Kotlin을 사용한 IAM 예제
- AWS IoT SDK for Kotlin을 사용한 예제
- AWS IoT data SDK for Kotlin을 사용한 예제

- SDK for Kotlin을 사용한 Amazon Keyspaces 예제
- AWS KMS SDK for Kotlin을 사용한 예제
- SDK for Kotlin을 사용한 Lambda 예제
- SDK for Kotlin을 사용한 MediaConvert 예제
- SDK for Kotlin을 사용한 Amazon Pinpoint 예제
- SDK for Kotlin을 사용한 Amazon RDS 예제
- SDK for Kotlin을 사용한 Amazon RDS Data Service 예제
- SDK for Kotlin을 사용한 Amazon Redshift 예제
- SDK for Kotlin을 사용한 Amazon Rekognition 예제
- SDK for Kotlin을 사용한 Route 53 도메인 등록 예제
- SDK for Kotlin을 사용한 Amazon S3 예제
- SDK for Kotlin을 사용한 SageMaker AI 예제
- SDK for Kotlin을 사용한 Secrets Manager 예제
- SDK for Kotlin을 사용한 Amazon SES 예제
- SDK for Kotlin을 사용한 Amazon SNS 예제
- SDK for Kotlin을 사용한 Amazon SQS 예제
- SDK for Kotlin을 사용한 Step Functions의 예제
- 지원 SDK for Kotlin을 사용한 예제
- SDK for Kotlin을 사용한 Amazon Translate 예제

SDK for Kotlin을 사용한 API Gateway 예제

다음 코드 예제에서는 API Gateway와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시나리오

시나리오

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예시에서 사용되는 서비스

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

SDK for Kotlin을 사용한 Aurora 예제

다음 코드 예제에서는 AWS SDK for Kotlin을 Aurora와 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 전체 소스 코드에 대한 링크가 포함되어 있습니다.

주제

- 기본 사항
- 작업
- 시나리오

기본 사항

기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용자 지정 Aurora DB 클러스터 파라미터 그룹을 만들고 파라미터 값을 설정합니다.
- 파라미터 그룹을 사용하는 DB 클러스터를 생성합니다.
- 데이터베이스가 포함된 DB 인스턴스를 생성합니다.
- DB 클러스터의 스냅샷을 만든 다음, 리소스를 정리합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:
```

```
https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html

This Kotlin example performs the following tasks:

1. Returns a list of the available DB engines.
2. Creates a custom DB parameter group.
3. Gets the parameter groups.
4. Gets the parameters in the group.
5. Modifies the auto_increment_increment parameter.
6. Displays the updated parameter value.
7. Gets a list of allowed engine versions.
8. Creates an Aurora DB cluster database.
9. Waits for DB instance to be ready.
10. Gets a list of instance classes available for the selected engine.
11. Creates a database instance in the cluster.
12. Waits for the database instance in the cluster to be ready.
13. Creates a snapshot.
14. Waits for DB snapshot to be ready.
15. Deletes the DB instance.
16. Deletes the DB cluster.
17. Deletes the DB cluster group.
 */

var slTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbName> <dbSnapshotIdentifier> <secretName>
        Where:
            dbClusterGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceClusterIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
 the database credentials.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
```

```
    }

    val dbClusterGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceClusterIdentifier = args[2]
    val dbInstanceIdentifier = args[3]
    val dbName = args[4]
    val dbSnapshotIdentifier = args[5]
    val secretName = args[6]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeAuroraDBEngines()

    println("2. Create a custom parameter group")
    createDBClusterParameterGroup(dbClusterGroupName, dbParameterGroupFamily)

    println("3. Get the parameter group")
    describeDbClusterParameterGroups(dbClusterGroupName)

    println("4. Get the parameters in the group")
    describeDbClusterParameters(dbClusterGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBClusterParas(dbClusterGroupName)

    println("6. Display the updated parameter value")
    describeDbClusterParameters(dbClusterGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedClusterEngines(dbParameterGroupFamily)

    println("8. Create an Aurora DB cluster database")
    val arnClusterVal = createDBCluster(dbClusterGroupName, dbName,
dbInstanceClusterIdentifier, username, userPassword)
    println("The ARN of the cluster is $arnClusterVal")

    println("9. Wait for DB instance to be ready")
    waitForClusterInstanceReady(dbInstanceClusterIdentifier)
```

```
    println("10. Get a list of instance classes available for the selected engine")
    val instanceClass = getListInstanceClasses()

    println("11. Create a database instance in the cluster.")
    val clusterDBARN = createDBInstanceCluster(dbInstanceIdentifier,
 dbInstanceClusterIdentifier, instanceClass)
    println("The ARN of the database is $clusterDBARN")

    println("12. Wait for DB instance to be ready")
    waitDBAuroraInstanceReady(dbInstanceIdentifier)

    println("13. Create a snapshot")
    createDBClusterSnapshot(dbInstanceClusterIdentifier, dbSnapshotIdentifier)

    println("14. Wait for DB snapshot to be ready")
    waitSnapshotReady(dbSnapshotIdentifier, dbInstanceClusterIdentifier)

    println("15. Delete the DB instance")
    deleteDBInstance(dbInstanceIdentifier)

    println("16. Delete the DB cluster")
    deleteCluster(dbInstanceClusterIdentifier)

    println("17. Delete the DB cluster group")
    if (clusterDBARN != null) {
        deleteDBClusterGroup(dbClusterGroupName, clusterDBARN)
    }
    println("The Scenario has successfully completed.")
}

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

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
```

```
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}

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

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

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

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

suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
```

```
            }
        }
    }
    println("The Snapshot is available!")
}

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

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

suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
```

```
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}

suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

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

suspend fun getListInstanceClasses(): String {
    val optionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "aurora-mysql"
            maxRecords = 20
        }
    var instanceClass = ""
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeOrderableDbInstanceOptions(optionsRequest)
        response.orderableDbInstanceOptions?.forEach { instanceOption ->
            instanceClass = instanceOption.dbInstanceClass.toString()
            println("The instance class is ${instanceOption.dbInstanceClass}")
            println("The engine version is ${instanceOption.engineVersion}")
        }
    }
    return instanceClass
}
```

```
// Waits until the database instance is available.
suspend fun waitForClusterInstanceReady(dbClusterIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbClustersRequest {
            dbClusterIdentifier = dbClusterIdentifierVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbClusters(instanceRequest)
            response.dbClusters?.forEach { cluster ->
                instanceReadyStr = cluster.status.toString()
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database cluster is available!")
}

suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
```

```
        }

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

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

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
```

```
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}

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

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

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

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}

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

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

suspend fun describeAuroraDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            engine = "aurora-mysql"
            defaultOnly = true
            maxRecords = 20
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
```

```
        response.dbEngineVersions?.forEach { engineOb ->
            println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}")
            println("The name of the database engine ${engineOb.engine}")
            println("The version number of the database engine
 ${engineOb.engineVersion}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

작업

`CreateDBCluster`

다음 코드 예시에서는 `CreateDBCluster`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateDBCluster를 참조하십시오.

`CreateDBClusterParameterGroup`

다음 코드 예시에서는 `CreateDBClusterParameterGroup`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateDBCluster를 참조하십시오.

`CreateDBClusterSnapshot`

다음 코드 예시에서는 `CreateDBClusterSnapshot`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateDBClusterSnapshot을 참조하십시오.

`CreateDBInstance`

다음 코드 예시에서는 `CreateDBInstance`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateDBInstance를 참조하십시오.

`DeleteDBCluster`

다음 코드 예시에서는 `DeleteDBCluster`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteDBCluster를 참조하십시오.

`DeleteDBClusterParameterGroup`

다음 코드 예시에서는 `DeleteDBClusterParameterGroup`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
```

```
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteDBClusterParameterGroup을 참조하십시오.

`DeleteDBInstance`

다음 코드 예시에서는 `DeleteDBInstance`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
```

```
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteDBInstance를 참조하십시오.

`DescribeDBClusterParameterGroups`

다음 코드 예시에서는 `DescribeDBClusterParameterGroups`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeDBClusterParameterGroups를 참조하십시오.

`DescribeDBClusterParameters`

다음 코드 예시에서는 `DescribeDBClusterParameters`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is ${para.description}")
```

```
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeDBClusterParameters를 참조하십시오.

`DescribeDBClusterSnapshots`

다음 코드 예시에서는 `DescribeDBClusterSnapshots`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
```

```
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeDBClusterSnapshots를 참조하십시오.

`DescribeDBClusters`

다음 코드 예시에서는 `DescribeDBClusters`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeDBClusters를 참조하십시오.

`DescribeDBEngineVersions`

다음 코드 예시에서는 `DescribeDBEngineVersions`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeDBEngineVersions을 참조하십시오.

`DescribeDBInstances`

다음 코드 예시에서는 `DescribeDBInstances`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }
```

```
    var endpoint = ""
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeDBInstances를 참조하십시오.

`ModifyDBClusterParameterGroup`

다음 코드 예시에서는 `ModifyDBClusterParameterGroup`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }
```

```
    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ModifyDBClusterParameterGroup을 참조하십시오.

시나리오

Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제는 Amazon Aurora Serverless 데이터베이스의 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 전송하는 웹 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon RDS 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

Amazon Aurora Serverless 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 Spring REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 GitHub에서 전체 예제를 참조하세요.

이 예시에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스

- Amazon SES

SDK for Kotlin을 사용한 Auto Scaling 예제

다음 코드 예제에서는 Auto Scaling과 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 전체 소스 코드에 대한 링크가 포함되어 있습니다.

주제

- 기본 사항
- 작업

기본 사항

기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 시작 템플릿과 가용 영역이 있는 Amazon EC2 Auto Scaling 그룹을 생성하고 실행 중인 인스턴스에 대한 정보를 가져옵니다.
- Amazon CloudWatch 지표 수집 활성화
- 그룹의 원하는 용량을 업데이트하고 인스턴스가 시작될 때까지 기다립니다.
- 그룹에서 인스턴스를 종료합니다.
- 사용자 요청 및 용량 변경에 따라 발생하는 조정 활동을 나열합니다.
- CloudWatch 지표에 대한 통계를 가져온 다음 리소스를 정리합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>

    Where:
        groupName - The name of the Auto Scaling group.
        launchTemplateName - The name of the launch template.
        serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-linked
 role that the Auto Scaling group uses.
        vpcZoneId - A subnet Id for a virtual private cloud (VPC) where instances in
 the Auto Scaling group can be created.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val groupName = args[0]
    val launchTemplateName = args[1]
    val serviceLinkedRoleARN = args[2]
    val vpcZoneId = args[3]

    println("**** Create an Auto Scaling group named $groupName")
    createAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN,
 vpcZoneId)

    println("Wait 1 min for the resources, including the instance. Otherwise, an
 empty instance Id is returned")
    delay(60000)

    val instanceId = getSpecificAutoScaling(groupName)
    if (instanceId.compareTo("") == 0) {
```

```
        println("Error - no instance Id value")
        exitProcess(1)
    } else {
        println("The instance Id value is $instanceId")
    }

    println("**** Describe Auto Scaling with the Id value $instanceId")
    describeAutoScalingInstance(instanceId)

    println("**** Enable metrics collection $instanceId")
    enableMetricsCollection(groupName)

    println("**** Update an Auto Scaling group to maximum size of 3")
    updateAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN)

    println("**** Describe all Auto Scaling groups to show the current state of the
 groups")
    describeAutoScalingGroups(groupName)

    println("**** Describe account details")
    describeAccountLimits()

    println("Wait 1 min for the resources, including the instance. Otherwise, an
 empty instance Id is returned")
    delay(60000)

    println("**** Set desired capacity to 2")
    setDesiredCapacity(groupName)

    println("**** Get the two instance Id values and state")
    getAutoScalingGroups(groupName)

    println("**** List the scaling activities that have occurred for the group")
    describeScalingActivities(groupName)

    println("**** Terminate an instance in the Auto Scaling group")
    terminateInstanceInAutoScalingGroup(instanceId)

    println("**** Stop the metrics collection")
    disableMetricsCollection(groupName)

    println("**** Delete the Auto Scaling group")
    deleteSpecificAutoScalingGroup(groupName)
}
```

```
suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}

suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}

suspend fun describeScalingActivities(groupName: String?) {
    val scalingActivitiesRequest =
        DescribeScalingActivitiesRequest {
            autoScalingGroupName = groupName
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeScalingActivities(scalingActivitiesRequest)
        response.activities?.forEach { activity ->
            println("The activity Id is ${activity.activityId}")
            println("The activity details are ${activity.details}")
        }
```

```
    }
}

suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}

suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}

suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
```

```
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
    }
}

suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }
```

```
    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}

suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}

suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}

suspend fun getSpecificAutoScaling(groupName: String): String {
    var instanceId = ""
```

```
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")

            group.instances?.forEach { instance ->
                instanceId = instance.instanceId.toString()
            }
        }
    }
    return instanceId
}

suspend fun describeAccountLimits() {
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAccountLimits(DescribeAccountLimitsRequest {})
        println("The max number of Auto Scaling groups is
 ${response.maxNumberOfAutoScalingGroups}")
        println("The current number of Auto Scaling groups is
 ${response.numberOfAutoScalingGroups}")
    }
}

suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}
```

```
suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

작업

`CreateAutoScalingGroup`

다음 코드 예시에서는 `CreateAutoScalingGroup`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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!")
    }
}
```

- API에 대한 자세한 설명은 Kotlin API를 위한AWS SDK 참조 문서에서 CreateAutoScalingGroup을 참조하세요.

`DeleteAutoScalingGroup`

다음 코드 예시에서는 `DeleteAutoScalingGroup`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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")
    }
}
```

- API에 대한 자세한 설명은 Kotlin API를 위한AWS SDK 참조 문서의 DeleteAutoScalingGroup을 참조하세요.

`DescribeAutoScalingGroups`

다음 코드 예시에서는 `DescribeAutoScalingGroups`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeAutoScalingGroups을 참조하십시오.

`DescribeAutoScalingInstances`

다음 코드 예시에서는 `DescribeAutoScalingInstances`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeAutoScalingInstances를 참조하십시오.

`DescribeScalingActivities`

다음 코드 예시에서는 `DescribeScalingActivities`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeAutoScalingGroups(groupName: String) {
```

```
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeScalingActivities를 참조하십시오.

`DisableMetricsCollection`

다음 코드 예시에서는 `DisableMetricsCollection`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
```

```
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DisableMetricsCollection을 참조하십시오.

`EnableMetricsCollection`

다음 코드 예시에서는 `EnableMetricsCollection`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 EnableMetricsCollection을 참조하십시오.

`SetDesiredCapacity`

다음 코드 예시에서는 `SetDesiredCapacity`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 SetDesiredCapacity를 참조하십시오.

`TerminateInstanceInAutoScalingGroup`

다음 코드 예시에서는 `TerminateInstanceInAutoScalingGroup`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
```

```
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 TerminateInstanceInAutoScalingGroup을 참조하십시오.

`UpdateAutoScalingGroup`

다음 코드 예시에서는 `UpdateAutoScalingGroup`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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")
    }
}
```

- API에 대한 자세한 설명은 Kotlin API를 위한AWS SDK 참조 문서의 UpdateAutoScalingGroup을 참조하세요.

SDK for Kotlin를 사용한 Amazon Bedrock 예시

다음 코드 예제에서는 Amazon Bedrock과 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`ListFoundationModels`

다음 코드 예시에서는 `ListFoundationModels`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

사용 가능한 Amazon Bedrock 기초 모델을 나열하세요.

```
suspend fun listFoundationModels(): List<FoundationModelSummary>? {
    BedrockClient { region = "us-east-1" }.use { bedrockClient ->
        val response =
 bedrockClient.listFoundationModels(ListFoundationModelsRequest {})
        response.modelSummaries?.forEach { model ->
            println("========================================")
            println(" Model ID: ${model.modelId}")
            println("----------------------------------------")
            println(" Name: ${model.modelName}")
            println(" Provider: ${model.providerName}")
            println(" Input modalities: ${model.inputModalities}")
            println(" Output modalities: ${model.outputModalities}")
            println(" Supported customizations: ${model.customizationsSupported}")
            println(" Supported inference types: ${model.inferenceTypesSupported}")
            println("----------------------------------------\n")
        }
        return response.modelSummaries
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListFoundationModels를 참조하세요.

SDK for Kotlin을 사용한 CloudWatch 예제

다음 코드 예제에서는 CloudWatch와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시작

Hello CloudWatch

다음 코드 예제에서는 CloudWatch를 사용하여 시작하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
           <namespace>
        Where:
           namespace - The namespace to filter against (for example, AWS/EC2).
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val namespace = args[0]
```

```
    listAllMets(namespace)
}

suspend fun listAllMets(namespaceVal: String?) {
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listMetricsPaginated(request)
            .transform { it.metrics?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.metricName}")
                println("Namespace is ${obj.namespace}")
            }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListMetrics를 참조하십시오.

주제

- 기본 사항
- 작업

기본 사항

기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- CloudWatch 네임스페이스 및 지표를 나열합니다.
- 지표 및 예상 청구에 대한 통계를 가져옵니다.
- 대시보드를 생성하고 업데이트합니다.
- 데이터를 생성하여 지표에 추가합니다.
- 경보를 생성하고 트리거한 다음 경보 기록을 봅니다.
- 이상 탐지기를 추가합니다.

- 지표 이미지를 가져온 다음 리소스를 정리합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

CloudWatch 기능을 보여주는 대화형 시나리오를 실행합니다.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 To enable billing metrics and statistics for this example, make sure billing alerts
 are enabled for your account:
 https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics

 This Kotlin code example performs the following tasks:

 1. List available namespaces from Amazon CloudWatch. Select a namespace from the
 list.
 2. List available metrics within the selected namespace.
 3. Get statistics for the selected metric over the last day.
 4. Get CloudWatch estimated billing for the last week.
 5. Create a new CloudWatch dashboard with metrics.
 6. List dashboards using a paginator.
 7. Create a new custom metric by adding data for it.
 8. Add the custom metric to the dashboard.
 9. Create an alarm for the custom metric.
 10. Describe current alarms.
 11. Get current data for the new custom metric.
 12. Push data into the custom metric to trigger the alarm.
 13. Check the alarm state using the action DescribeAlarmsForMetric.
 14. Get alarm history for the new alarm.
```

```
 15. Add an anomaly detector for the custom metric.
 16. Describe current anomaly detectors.
 17. Get a metric image for the custom metric.
 18. Clean up the Amazon CloudWatch resources.
 */

val DASHES: String? = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <myDate> <costDateWeek> <dashboardName> <dashboardJson> <dashboardAdd>
 <settings> <metricImage>

        Where:
            myDate - The start date to use to get metric statistics. (For example,
 2023-01-11T18:35:24.00Z.)
            costDateWeek - The start date to use to get AWS Billing and Cost
 Management statistics. (For example, 2023-01-11T18:35:24.00Z.)
            dashboardName - The name of the dashboard to create.
            dashboardJson - The location of a JSON file to use to create a
 dashboard. (See Readme file.)
            dashboardAdd - The location of a JSON file to use to update a dashboard.
 (See Readme file.)
            settings - The location of a JSON file from which various values are
 read. (See Readme file.)
            metricImage - The location of a BMP file that is used to create a
 graph.
    """

    if (args.size != 7) {
        println(usage)
        System.exit(1)
    }

    val myDate = args[0]
    val costDateWeek = args[1]
    val dashboardName = args[2]
    val dashboardJson = args[3]
    val dashboardAdd = args[4]
    val settings = args[5]
    var metricImage = args[6]
    val dataPoint = "10.0".toDouble()
    val inOb = Scanner(System.`in`)
```

```
    println(DASHES)
    println("Welcome to the Amazon CloudWatch example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List at least five available unique namespaces from Amazon
CloudWatch. Select a CloudWatch namespace from the list.")
    val list: ArrayList<String> = listNameSpaces()
    for (z in 0..4) {
        println("    ${z + 1}. ${list[z]}")
    }

    var selectedNamespace: String
    var selectedMetrics = ""
    var num = inOb.nextLine().toInt()
    println("You selected $num")

    if (1 <= num && num <= 5) {
        selectedNamespace = list[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $selectedNamespace")
    println(DASHES)

    println(DASHES)
    println("2. List available metrics within the selected namespace and select one
from the list.")
    val metList = listMets(selectedNamespace)
    for (z in 0..4) {
        println("    ${ z + 1}. ${metList?.get(z)}")
    }
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        selectedMetrics = metList![num - 1]
    } else {
        println("You did not select a valid option.")
        System.exit(1)
    }
    println("You selected $selectedMetrics")
    val myDimension = getSpecificMet(selectedNamespace)
    if (myDimension == null) {
```

```
        println("Error - Dimension is null")
        exitProcess(1)
    }
    println(DASHES)

    println(DASHES)
    println("3. Get statistics for the selected metric over the last day.")
    val metricOption: String
    val statTypes = ArrayList<String>()
    statTypes.add("SampleCount")
    statTypes.add("Average")
    statTypes.add("Sum")
    statTypes.add("Minimum")
    statTypes.add("Maximum")

    for (t in 0..4) {
        println("    ${t + 1}. ${statTypes[t]}")
    }
    println("Select a metric statistic by entering a number from the preceding
list:")
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        metricOption = statTypes[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $metricOption")
    getAndDisplayMetricStatistics(selectedNamespace, selectedMetrics, metricOption,
myDate, myDimension)
    println(DASHES)

    println(DASHES)
    println("4. Get CloudWatch estimated billing for the last week.")
    getMetricStatistics(costDateWeek)
    println(DASHES)

    println(DASHES)
    println("5. Create a new CloudWatch dashboard with metrics.")
    createDashboardWithMetrics(dashboardName, dashboardJson)
    println(DASHES)

    println(DASHES)
    println("6. List dashboards using a paginator.")
```

```
    listDashboards()
    println(DASHES)

    println(DASHES)
    println("7. Create a new custom metric by adding data to it.")
    createNewCustomMetric(dataPoint)
    println(DASHES)

    println(DASHES)
    println("8. Add an additional metric to the dashboard.")
    addMetricToDashboard(dashboardAdd, dashboardName)
    println(DASHES)

    println(DASHES)
    println("9. Create an alarm for the custom metric.")
    val alarmName: String = createAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("10. Describe 10 current alarms.")
    describeAlarms()
    println(DASHES)

    println(DASHES)
    println("11. Get current data for the new custom metric.")
    getCustomMetricData(settings)
    println(DASHES)

    println(DASHES)
    println("12. Push data into the custom metric to trigger the alarm.")
    addMetricDataForAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("13. Check the alarm state using the action DescribeAlarmsForMetric.")
    checkForMetricAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("14. Get alarm history for the new alarm.")
    getAlarmHistory(settings, myDate)
    println(DASHES)

    println(DASHES)
```

```
    println("15. Add an anomaly detector for the custom metric.")
    addAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("16. Describe current anomaly detectors.")
    describeAnomalyDetectors(settings)
    println(DASHES)

    println(DASHES)
    println("17. Get a metric image for the custom metric.")
    getAndOpenMetricImage(metricImage)
    println(DASHES)

    println(DASHES)
    println("18. Clean up the Amazon CloudWatch resources.")
    deleteDashboard(dashboardName)
    deleteAlarm(alarmName)
    deleteAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("The Amazon CloudWatch example scenario is complete.")
    println(DASHES)
}

suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val request =
        DeleteAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }
```

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

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

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

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

suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
```

```
            "USD"
            ]
        ]
        }"""

    val imageRequest =
        GetMetricWidgetImageRequest {
            metricWidget = myJSON
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}

suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest =
        DescribeAnomalyDetectorsRequest {
            maxResults = 10
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}

suspend fun addAnomalyDetector(fileName: String?) {
```

```
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val anomalyDetectorRequest =
        PutAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

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

suspend fun getAlarmHistory(
    fileName: String,
    date: String,
) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest =
        DescribeAlarmHistoryRequest {
            startDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDateVal)
            alarmName = alarmNameVal
```

```
            historyItemType = HistoryItemType.Action
        }

    CloudWatchClient {
        credentialsProvider = EnvironmentCredentialsProvider()
        region = "us-east-1"
    }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}

suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest =
        DescribeAlarmsForMetricRequest {
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
```

```
            println(".")
        }
        if (!hasAlarm) {
            println("No Alarm state found for $customMetricName after 10 retries.")
        } else {
            println("Alarm state found for $customMetricName.")
        }
    }
}

suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1001.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val datum2 =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1002.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)
```

```
    val request =
        PutMetricDataRequest {
            namespace = customMetricNamespace
            metricData = metricDataList
        }

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

suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 =
        nowDate.plus(hours, ChronoUnit.HOURS).plus(
            minutes,
            ChronoUnit.MINUTES,
        )

    val met =
        Metric {
            metricName = customMetricName
            namespace = customMetricNamespace
        }

    val metStat =
        MetricStat {
            stat = "Maximum"
            period = 1
            metric = met
        }

    val dataQUery =
```

```
        MetricDataQuery {
            metricStat = metStat
            id = "foo2"
            returnData = true
        }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq =
        GetMetricDataRequest {
            maxDatapoints = 10
            scanBy = ScanBy.TimestampDescending
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(nowDate)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(date2)
            metricDataQueries = dq
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}

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

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
```

```
        }
    }
}

suspend fun createAlarm(fileName: String): String {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode: JsonNode = ObjectMapper().readTree(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val emailTopic = rootNode.findValue("emailTopic").asText()
    val accountId = rootNode.findValue("accountId").asText()
    val region2 = rootNode.findValue("region").asText()

    // Create a List for alarm actions.
    val alarmActionObs: MutableList<String> = ArrayList()
    alarmActionObs.add("arn:aws:sns:$region2:$accountId:$emailTopic")
    val alarmRequest =
        PutMetricAlarmRequest {
            alarmActions = alarmActionObs
            alarmDescription = "Example metric alarm"
            alarmName = alarmNameVal
            comparisonOperator = ComparisonOperator.GreaterThanOrEqualToThreshold
            threshold = 100.00
            metricName = customMetricName
            namespace = customMetricNamespace
            evaluationPeriods = 1
            period = 10
            statistic = Statistic.Maximum
            datapointsToAlarm = 1
            treatMissingData = "ignore"
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(alarmRequest)
        println("$alarmNameVal was successfully created!")
        return alarmNameVal
    }
}

suspend fun addMetricToDashboard(
    fileNameVal: String,
    dashboardNameVal: String,
```

```
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully updated.")
    }
}

suspend fun createNewCustomMetric(dataPoint: Double) {
    val dimension =
        Dimension {
            name = "UNIQUE_PAGES"
            value = "URLS"
        }

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = "PAGES_VISITED"
            unit = StandardUnit.None
            value = dataPoint
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
            dimensions = listOf(dimension)
        }

    val request =
        PutMetricDataRequest {
            namespace = "SITE/TRAFFIC"
            metricData = listOf(datum)
        }

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

```
    }
}

suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}

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

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}

fun readFileAsString(file: String): String =
 String(Files.readAllBytes(Paths.get(file)))
```

```
suspend fun getMetricStatistics(costDateWeek: String?) {
    val start = Instant.parse(costDateWeek)
    val endDate = Instant.now()
    val dimension =
        Dimension {
            name = "Currency"
            value = "USD"
        }

    val dimensionList: MutableList<Dimension> = ArrayList()
    dimensionList.add(dimension)

    val statisticsRequest =
        GetMetricStatisticsRequest {
            metricName = "EstimatedCharges"
            namespace = "AWS/Billing"
            dimensions = dimensionList
            statistics = listOf(Statistic.Maximum)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            period = 86400
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data: List<Datapoint>? = response.datapoints
        if (data != null) {
            if (!data.isEmpty()) {
                for (datapoint in data) {
                    println("Timestamp:  ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

suspend fun getAndDisplayMetricStatistics(
    nameSpaceVal: String,
```

```
    metVal: String,
    metricOption: String,
    date: String,
    myDimension: Dimension,
) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest =
        GetMetricStatisticsRequest {
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            dimensions = listOf(myDimension)
            metricName = metVal
            namespace = nameSpaceVal
            period = 86400
            statistics = listOf(Statistic.fromValue(metricOption))
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

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

```
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}

suspend fun getSpecificMet(namespaceVal: String?): Dimension? {
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(request)
        val myList = response.metrics
        if (myList != null) {
            return myList[0].dimensions?.get(0)
        }
    }
    return null
}

suspend fun listNameSpaces(): ArrayList<String> {
    val nameSpaceList = ArrayList<String>()
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(ListMetricsRequest {})
        response.metrics?.forEach { metrics ->
            val data = metrics.namespace
            if (!nameSpaceList.contains(data)) {
                nameSpaceList.add(data!!)
            }
        }
    }
    return nameSpaceList
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.

- DeleteAlarms
- DeleteAnomalyDetector
- DeleteDashboards
- DescribeAlarmHistory
- DescribeAlarms
- DescribeAlarmsForMetric
- DescribeAnomalyDetectors
- GetMetricData
- GetMetricStatistics
- GetMetricWidgetImage
- ListMetrics
- PutAnomalyDetector
- PutDashboard
- PutMetricAlarm
- PutMetricData

작업

`DeleteAlarms`

다음 코드 예시에서는 `DeleteAlarms`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteAlarms를 참조하십시오.

`DeleteAnomalyDetector`

다음 코드 예시에서는 `DeleteAnomalyDetector`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val request =
        DeleteAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }
```

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteAnomalyDetector를 참조하십시오.

`DeleteDashboards`

다음 코드 예시에서는 `DeleteDashboards`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteDashboards를 참조하십시오.

`DescribeAlarmHistory`

다음 코드 예시에서는 `DescribeAlarmHistory`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getAlarmHistory(
    fileName: String,
    date: String,
) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest =
        DescribeAlarmHistoryRequest {
            startDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDateVal)
            alarmName = alarmNameVal
            historyItemType = HistoryItemType.Action
        }

    CloudWatchClient {
        credentialsProvider = EnvironmentCredentialsProvider()
        region = "us-east-1"
    }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
```

```
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeAlarmHistory를 참조하십시오.

`DescribeAlarms`

다음 코드 예시에서는 `DescribeAlarms`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeAlarms를 참조하십시오.

`DescribeAlarmsForMetric`

다음 코드 예시에서는 `DescribeAlarmsForMetric`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest =
        DescribeAlarmsForMetricRequest {
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) {
            println("No Alarm state found for $customMetricName after 10 retries.")
        } else {
            println("Alarm state found for $customMetricName.")
```

```
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeAlarmsForMetric를 참조하십시오.

`DescribeAnomalyDetectors`

다음 코드 예시에서는 `DescribeAnomalyDetectors`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest =
        DescribeAnomalyDetectorsRequest {
            maxResults = 10
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeAnomalyDetectors를 참조하십시오.

`DisableAlarmActions`

다음 코드 예시에서는 `DisableAlarmActions`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DisableAlarmActions를 참조하십시오.

`EnableAlarmActions`

다음 코드 예시에서는 `EnableAlarmActions`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 EnableAlarmActions를 참조하십시오.

`GetMetricData`

다음 코드 예시에서는 `GetMetricData`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 =
        nowDate.plus(hours, ChronoUnit.HOURS).plus(
            minutes,
```

```
            ChronoUnit.MINUTES,
        )

    val met =
        Metric {
            metricName = customMetricName
            namespace = customMetricNamespace
        }

    val metStat =
        MetricStat {
            stat = "Maximum"
            period = 1
            metric = met
        }

    val dataQUery =
        MetricDataQuery {
            metricStat = metStat
            id = "foo2"
            returnData = true
        }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq =
        GetMetricDataRequest {
            maxDatapoints = 10
            scanBy = ScanBy.TimestampDescending
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(nowDate)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(date2)
            metricDataQueries = dq
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
```

```
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetMetricData를 참조하십시오.

`GetMetricStatistics`

다음 코드 예시에서는 `GetMetricStatistics`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getAndDisplayMetricStatistics(
    nameSpaceVal: String,
    metVal: String,
    metricOption: String,
    date: String,
    myDimension: Dimension,
) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest =
        GetMetricStatisticsRequest {
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            dimensions = listOf(myDimension)
            metricName = metVal
            namespace = nameSpaceVal
            period = 86400
            statistics = listOf(Statistic.fromValue(metricOption))
        }
```

```
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetMetricStatistics를 참조하십시오.

`GetMetricWidgetImage`

다음 코드 예시에서는 `GetMetricWidgetImage`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
```

```
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
        }"""

    val imageRequest =
        GetMetricWidgetImageRequest {
            metricWidget = myJSON
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetMetricWidgetImage를 참조하십시오.

`ListDashboards`

다음 코드 예시에서는 `ListDashboards`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listDashboards() {
```

```
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListDashboards를 참조하십시오.

`ListMetrics`

다음 코드 예시에서는 `ListMetrics`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
```

```
    return metList
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListMetrics를 참조하십시오.

`PutAnomalyDetector`

다음 코드 예시에서는 `PutAnomalyDetector`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val anomalyDetectorRequest =
        PutAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

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

```
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 PutAnomalyDetector를 참조하십시오.

`PutDashboard`

다음 코드 예시에서는 `PutDashboard`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
```

```
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 PutDashboard를 참조하십시오.

`PutMetricAlarm`

다음 코드 예시에서는 `PutMetricAlarm`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun putMetricAlarm(
    alarmNameVal: String,
    instanceIdVal: String,
) {
    val dimensionOb =
        Dimension {
            name = "InstanceId"
            value = instanceIdVal
        }

    val request =
        PutMetricAlarmRequest {
            alarmName = alarmNameVal
            comparisonOperator = ComparisonOperator.GreaterThanThreshold
            evaluationPeriods = 1
            metricName = "CPUUtilization"
            namespace = "AWS/EC2"
            period = 60
            statistic = Statistic.fromValue("Average")
            threshold = 70.0
            actionsEnabled = false
            alarmDescription = "An Alarm created by the Kotlin SDK when server CPU
 utilization exceeds 70%"
            unit = StandardUnit.fromValue("Seconds")
```

```
            dimensions = listOf(dimensionOb)
        }

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 PutMetricAlarm를 참조하십시오.

`PutMetricData`

다음 코드 예시에서는 `PutMetricData`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1001.00
            timestamp =
                aws.smithy.kotlin.runtime.time
```

```
                    .Instant(instant)
        }

    val datum2 =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1002.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)

    val request =
        PutMetricDataRequest {
            namespace = customMetricNamespace
            metricData = metricDataList
        }

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 PutMetricData를 참조하십시오.

SDK for Kotlin을 사용한 CloudWatch Logs 예제

다음 코드 예제에서는 CloudWatch Logs와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`DeleteSubscriptionFilter`

다음 코드 예시에서는 `DeleteSubscriptionFilter`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteSubscriptionFilter를 참조하십시오.

`DescribeSubscriptionFilters`

다음 코드 예시에서는 `DescribeSubscriptionFilters`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

    CloudWatchLogsClient { region = "us-west-2" }.use { cwlClient ->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name  ${filter.filterName} pattern
 ${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeSubscriptionFilters를 참조하십시오.

`StartLiveTail`

다음 코드 예시에서는 `StartLiveTail`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

필수 파일을 포함합니다.

```
import aws.sdk.kotlin.services.cloudwatchlogs.CloudWatchLogsClient
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailRequest
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailResponseStream
import kotlinx.coroutines.flow.takeWhile
```

Live Tail 세션을 시작합니다.

```
    val client = CloudWatchLogsClient.fromEnvironment()

    val request = StartLiveTailRequest {
        logGroupIdentifiers = logGroupIdentifiersVal
        logStreamNames = logStreamNamesVal
        logEventFilterPattern = logEventFilterPatternVal
    }

    val startTime = System.currentTimeMillis()

    try {
        client.startLiveTail(request) { response ->
            val stream = response.responseStream
            if (stream != null) {
                /* Set a timeout to unsubcribe from the flow. This will:
                * 1). Close the stream
                * 2). Stop the Live Tail session
                */
                stream.takeWhile { System.currentTimeMillis() - startTime <
10000 }.collect { value ->
                    if (value is StartLiveTailResponseStream.SessionStart) {
                        println(value.asSessionStart())
                    } else if (value is StartLiveTailResponseStream.SessionUpdate) {
                        for (e in value.asSessionUpdate().sessionResults!!) {
                            println(e)
                        }
                    } else {
                        throw IllegalArgumentException("Unknown event type")
                    }
                }
            } else {
                throw IllegalArgumentException("No response stream")
            }
        }
    } catch (e: Exception) {
        println("Exception occurred during StartLiveTail: $e")
        System.exit(1)
    }
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 StartLiveTail을 참조하세요.

SDK for Kotlin을 사용한 Amazon Cognito 자격 증명 공급자용 코드 예제

다음 코드 예제에서는 Amazon Cognito Identity Provider와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업
- 시나리오

작업

`AdminGetUser`

다음 코드 예시에서는 `AdminGetUser`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

```
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 AdminGetUser를 참조하십시오.

`AdminInitiateAuth`

다음 코드 예시에서는 `AdminInitiateAuth`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
```

```
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 AdminInitiateAuth를 참조하십시오.

`AdminRespondToAuthChallenge`

다음 코드 예시에서는 `AdminRespondToAuthChallenge`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
```

```
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 AdminRespondToAuthChallenge를 참조하십시오.

`AssociateSoftwareToken`

다음 코드 예시에서는 `AssociateSoftwareToken`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
```

```
        println(secretCode)
        return tokenResponse.session
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 AssociateSoftwareToken을 참조하십시오.

`ConfirmSignUp`

다음 코드 예시에서는 `ConfirmSignUp`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ConfirmSignUp을 참조하십시오.

`ListUsers`

다음 코드 예시에서는 `ListUsers`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listAllUsers(userPoolId: String) {
    val request =
        ListUsersRequest {
            this.userPoolId = userPoolId
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListUsers를 참조하십시오.

`ResendConfirmationCode`

다음 코드 예시에서는 `ResendConfirmationCode`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ResendConfirmationCode를 참조하십시오.

`SignUp`

다음 코드 예시에서는 `SignUp`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
```

```
            name = "email"
            value = emailVal
        }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 SignUp을 참조하십시오.

`VerifySoftwareToken`

다음 코드 예시에서는 `VerifySoftwareToken`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
```

```
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 VerifySoftwareToken을 참조하십시오.

시나리오

MFA가 필요한 사용자 풀에 사용자 가입시키기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용자 이름, 암호 및 이메일 주소로 사용자를 가입시키고 확인합니다.
- MFA 애플리케이션을 사용자와 연결하여 다중 인증을 설정합니다.
- 암호와 MFA 코드를 사용하여 로그인합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation:
```

```
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 TIP: To set up the required user pool, run the AWS Cloud Development
 Kit (AWS CDK) script provided in this GitHub repo at resources/cdk/
cognito_scenario_user_pool_with_mfa.

 This code example performs the following operations:

 1. Invokes the signUp method to sign up a user.
 2. Invokes the adminGetUser method to get the user's confirmation status.
 3. Invokes the ResendConfirmationCode method if the user requested another code.
 4. Invokes the confirmSignUp method.
 5. Invokes the initiateAuth to sign in. This results in being prompted to
 set up TOTP (time-based one-time password). (The response is “ChallengeName”:
 “MFA_SETUP”).
 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key.
 This can be used with Google Authenticator.
 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for MFA.
 8. Invokes the AdminInitiateAuth to sign in again. This results in being prompted
 to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <clientId> <poolId>
        Where:
            clientId - The app client Id value that you can get from the AWS CDK
 script.
            poolId - The pool Id that you can get from the AWS CDK script.
    """

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    val clientId = args[0]
    val poolId = args[1]

    // Use the console to get data from the user.
    println("*** Enter your use name")
    val inOb = Scanner(System.`in`)
```

```
    val userName = inOb.nextLine()
    println(userName)

    println("*** Enter your password")
    val password: String = inOb.nextLine()

    println("*** Enter your email")
    val email = inOb.nextLine()

    println("*** Signing up $userName")
    signUp(clientId, userName, password, email)

    println("*** Getting $userName in the user pool")
    getAdminUser(userName, poolId)

    println("*** Conformation code sent to $userName. Would you like to send a new
 code? (Yes/No)")
    val ans = inOb.nextLine()

    if (ans.compareTo("Yes") == 0) {
        println("*** Sending a new confirmation code")
        resendConfirmationCode(clientId, userName)
    }
    println("*** Enter the confirmation code that was emailed")
    val code = inOb.nextLine()
    confirmSignUp(clientId, code, userName)

    println("*** Rechecking the status of $userName in the user pool")
    getAdminUser(userName, poolId)

    val authResponse = checkAuthMethod(clientId, userName, password, poolId)
    val mySession = authResponse.session
    val newSession = getSecretForAppMFA(mySession)
    println("*** Enter the 6-digit code displayed in Google Authenticator")
    val myCode = inOb.nextLine()

    // Verify the TOTP and register for MFA.
    verifyTOTP(newSession, myCode)
    println("*** Re-enter a 6-digit code displayed in Google Authenticator")
    val mfaCode: String = inOb.nextLine()
    val authResponse1 = checkAuthMethod(clientId, userName, password, poolId)
    val session2 = authResponse1.session
    adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}
```

```
suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}

suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

```
// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
```

```
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}

suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}

suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}

suspend fun getAdminUser(
    userNameVal: String?,
```

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

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

suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
            name = "email"
            value = emailVal
        }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

SDK for Kotlin을 사용한 Amazon Comprehend 예제

다음 코드 예제에서는 Amazon Comprehend와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시나리오

시나리오

메시징 애플리케이션 생성

다음 코드 예제에서는 Amazon SQS를 사용하여 메시징 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon SQS API를 사용하여 메시지를 보내고 검색하는 Spring REST API를 개발하는 방법을 보여 줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하십시오.

이 예시에서 사용되는 서비스

- Amazon Comprehend
- Amazon SQS

SDK for Kotlin을 사용한 DynamoDB 예제

다음 코드 예제에서는 DynamoDB와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 전체 소스 코드에 대한 링크가 포함되어 있습니다.

주제

- 기본 사항
- 작업
- 시나리오

기본 사항

기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 영화 데이터를 저장할 수 있는 테이블을 생성합니다.

- 테이블에 하나의 영화를 추가하고 가져오고 업데이트합니다.
- 샘플 JSON 파일에서 테이블에 영화 데이터를 씁니다.
- 특정 연도에 개봉된 영화를 쿼리합니다.
- 특정 연도 범위 동안 개봉된 영화를 스캔합니다.
- 테이블에서 영화를 삭제한 다음, 테이블을 삭제합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

DynamoDB 테이블을 생성합니다.

```
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 provisionedVal =
        ProvisionedThroughput {
            readCapacityUnits = 10
            writeCapacityUnits = 10
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            provisionedThroughput = provisionedVal
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->

        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
 ${response.tableDescription?.tableArn}")
    }
}
```

헬퍼 함수를 생성하여 샘플 JSON 파일을 다운로드하고 추출합니다.

```
// Load data into the table.
suspend fun loadData(
    tableName: String,
    fileName: String,
) {
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode
```

```
    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}
```

테이블에서 항목을 가져옵니다.

```
suspend fun getMovie(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

전체 예제는 다음과 같습니다.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
          <fileName>

        Where:
           fileName - The path to the moviedata.json you can download from the
 Amazon DynamoDB Developer Guide.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }
```

```
    // Get the moviedata.json from the Amazon DynamoDB Developer Guide.
    val tableName = "Movies"
    val fileName = args[0]
    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 provisionedVal =
        ProvisionedThroughput {
            readCapacityUnits = 10
            writeCapacityUnits = 10
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            provisionedThroughput = provisionedVal
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->

        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
 ${response.tableDescription?.tableArn}")
    }
}

// Load data into the table.
suspend fun loadData(
    tableName: String,
    fileName: String,
) {
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
```

```
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}

suspend fun getMovie(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request =
        GetItemRequest {
            key = keyToGet
```

```
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}

suspend fun deletIssuesTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun queryMovieTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionAlias: String,
): Int {
    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = "year"

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.N("2013")

    val request =
        QueryRequest {
            tableName = tableNameVal
            keyConditionExpression = "$partitionAlias = :$partitionKeyName"
            expressionAttributeNames = attrNameAlias
            this.expressionAttributeValues = attrValues
        }
```

```
    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}

suspend fun scanMovies(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하세요.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

작업

`CreateTable`

다음 코드 예시에서는 `CreateTable`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 provisionedVal =
        ProvisionedThroughput {
            readCapacityUnits = 10
            writeCapacityUnits = 10
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef)
            keySchema = listOf(keySchemaVal)
            provisionedThroughput = provisionedVal
            tableName = tableNameVal
```

```
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        var tableArn: String
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        tableArn = response.tableDescription!!.tableArn.toString()
        println("Table $tableArn is ready")
        return tableArn
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateTable를 참조하세요.

`DeleteItem`

다음 코드 예시에서는 `DeleteItem`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteDynamoDBItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request =
        DeleteItemRequest {
            tableName = tableNameVal
            key = keyToGet
```

```
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteItem(request)
        println("Item with key matching $keyVal was deleted")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteItem를 참조하세요.

`DeleteTable`

다음 코드 예시에서는 `DeleteTable`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteDynamoDBTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteTable를 참조하세요.

`GetItem`

다음 코드 예시에서는 `GetItem`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getSpecificItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetItem를 참조하세요.

`ListTables`

다음 코드 예시에서는 `ListTables`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listAllTables() {
    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.listTables(ListTablesRequest {})
        response.tableNames?.forEach { tableName ->
            println("Table name is $tableName")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListTables를 참조하세요.

`PutItem`

다음 코드 예시에서는 `PutItem`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun putItemInTable(
    tableNameVal: String,
    key: String,
    keyVal: String,
    albumTitle: String,
    albumTitleValue: String,
    awards: String,
    awardVal: String,
```

```
    songTitle: String,
    songTitleVal: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()

    // Add all content to the table.
    itemValues[key] = AttributeValue.S(keyVal)
    itemValues[songTitle] = AttributeValue.S(songTitleVal)
    itemValues[albumTitle] = AttributeValue.S(albumTitleValue)
    itemValues[awards] = AttributeValue.S(awardVal)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println(" A new item was placed into $tableNameVal.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 PutItem를 참조하세요.

`Query`

다음 코드 예시에서는 `Query`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun queryDynTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionKeyVal: String,
```

```
    partitionAlias: String,
): Int {
    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = partitionKeyName

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.S(partitionKeyVal)

    val request =
        QueryRequest {
            tableName = tableNameVal
            keyConditionExpression = "$partitionAlias = :$partitionKeyName"
            expressionAttributeNames = attrNameAlias
            this.expressionAttributeValues = attrValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Query를 참조하세요.

Scan

다음 코드 예시에서는 Scan을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun scanItems(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
```

```
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Scan를 참조하세요.

`UpdateItem`

다음 코드 예시에서는 `UpdateItem`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun updateTableItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
    name: String,
    updateVal: String,
) {
    val itemKey = mutableMapOf<String, AttributeValue>()
    itemKey[keyName] = AttributeValue.S(keyVal)

    val updatedValues = mutableMapOf<String, AttributeValueUpdate>()
    updatedValues[name] =
        AttributeValueUpdate {
            value = AttributeValue.S(updateVal)
```

```
            action = AttributeAction.Put
        }

    val request =
        UpdateItemRequest {
            tableName = tableNameVal
            key = itemKey
            attributeUpdates = updatedValues
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.updateItem(request)
        println("Item in $tableNameVal was updated")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 UpdateItem를 참조하십시오.

시나리오

DynamoDB 테이블에 데이터를 제출하기 위한 앱 구축

다음 코드 예제는 Amazon DynamoDB 테이블에 데이터를 제출하고 사용자가 테이블을 업데이트할 때 알리는 애플리케이션을 빌드하는 방법을 보여줍니다.

SDK for Kotlin

Amazon DynamoDB Kotlin API를 사용하여 데이터를 제출하고 Amazon SNS Kotlin API를 사용하여 문자 메시지를 보내는 기본 Android 애플리케이션을 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon SNS

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예시에서 사용되는 서비스

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

DynamoDB 데이터를 추적하는 웹 애플리케이션 만들기

다음 코드 예제는 Amazon DynamoDB 테이블의 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 전송하는 웹 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon DynamoDB API를 사용하여 DynamoDB 작업 데이터를 추적하는 동적 웹 애플리케이션을 만드는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon SES

PartiQL 문 배치를 사용하여 테이블 쿼리

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 여러 SELECT 문을 실행하여 항목 배치를 가져옵니다.

- 여러 INSERT 문을 실행하여 항목 배치를 추가합니다.
- 여러 UPDATE 문을 실행하여 항목 배치를 업데이트합니다.
- 여러 DELETE 문을 실행하여 항목 배치를 삭제합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun main() {
    val ddb = DynamoDbClient { region = "us-east-1" }
    val tableName = "MoviesPartiQBatch"
    println("Creating an Amazon DynamoDB table named $tableName with a key named id
 and a sort key named title.")
    createTablePartiQLBatch(ddb, tableName, "year")
    putRecordBatch(ddb)
    updateTableItemBatchBatch(ddb)
    deleteItemsBatch(ddb)
    deleteTablePartiQLBatch(tableName)
}

suspend fun createTablePartiQLBatch(
    ddb: DynamoDbClient,
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }
```

```
    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val provisionedVal =
        ProvisionedThroughput {
            readCapacityUnits = 10
            writeCapacityUnits = 10
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            provisionedThroughput = provisionedVal
            tableName = tableNameVal
        }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists {
        // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}

suspend fun putRecordBatch(ddb: DynamoDbClient) {
    val sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?, 'title' : ?,
 'info' : ?}"

    // Create three movies to add to the Amazon DynamoDB table.
    val parametersMovie1 = mutableListOf<AttributeValue>()
    parametersMovie1.add(AttributeValue.N("2022"))
    parametersMovie1.add(AttributeValue.S("My Movie 1"))
    parametersMovie1.add(AttributeValue.S("No Information"))
```

```
    val statementRequestMovie1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie1
        }

    // Set data for Movie 2.
    val parametersMovie2 = mutableListOf<AttributeValue>()
    parametersMovie2.add(AttributeValue.N("2022"))
    parametersMovie2.add(AttributeValue.S("My Movie 2"))
    parametersMovie2.add(AttributeValue.S("No Information"))

    val statementRequestMovie2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie2
        }

    // Set data for Movie 3.
    val parametersMovie3 = mutableListOf<AttributeValue>()
    parametersMovie3.add(AttributeValue.N("2022"))
    parametersMovie3.add(AttributeValue.S("My Movie 3"))
    parametersMovie3.add(AttributeValue.S("No Information"))

    val statementRequestMovie3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestMovie1)
    myBatchStatementList.add(statementRequestMovie2)
    myBatchStatementList.add(statementRequestMovie3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }
    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: " + response.toString())
    println("Added new movies using a batch command.")
```

```
}

suspend fun updateTableItemBatchBatch(ddb: DynamoDbClient) {
    val sqlStatement =
        "UPDATE MoviesPartiQBatch SET info = 'directors\":[\"Merian C. Cooper\",
\"Ernest B. Schoedsack' where year=? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))
    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }

    // Update record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec2
        }

    // Update record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
```

```
        }

    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: $response")
    println("Updated three movies using a batch command.")
    println("Items were updated!")
}

suspend fun deleteItemsBatch(ddb: DynamoDbClient) {
    // Specify three records to delete.
    val sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))

    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }

    // Specify record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec2
        }

    // Specify record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
```

```
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }

    ddb.batchExecuteStatement(batchRequest)
    println("Deleted three movies using a batch command.")
}

suspend fun deleteTablePartiQLBatch(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 BatchExecuteStatement를 참조하십시오.

PartiQL을 사용하여 테이블 쿼리

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- SELECT 문을 실행하여 항목을 가져옵니다.
- INSERT 문을 실행하여 항목을 추가합니다.
- UPDATE 문을 실행하여 항목을 업데이트합니다.
- DELETE 문을 실행하여 항목을 삭제합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
          <fileName>

        Where:
           fileName - The path to the moviedata.json file You can download from the
 Amazon DynamoDB Developer Guide.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    val ddb = DynamoDbClient { region = "us-east-1" }
    val tableName = "MoviesPartiQ"

    // Get the moviedata.json from the Amazon DynamoDB Developer Guide.
    val fileName = args[0]
    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 provisionedVal =
        ProvisionedThroughput {
            readCapacityUnits = 10
            writeCapacityUnits = 10
        }

    val request =
        CreateTableRequest {
```

```
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            provisionedThroughput = provisionedVal
            tableName = tableNameVal
        }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists {
        // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}

suspend fun loadDataPartiQL(
    ddb: DynamoDbClient,
    fileName: String,
) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
 'info' : ?}"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode
    var t = 0

    while (iter.hasNext()) {
        if (t == 200) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
        parameters.add(AttributeValue.N(year.toString()))
        parameters.add(AttributeValue.S(title))
        parameters.add(AttributeValue.S(info))

        executeStatementPartiQL(ddb, sqlStatement, parameters)
        println("Added Movie $title")
```

```
        parameters.clear()
        t++
    }
}

suspend fun getMoviePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?"
    val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
    parameters.add(AttributeValue.N("2012"))
    parameters.add(AttributeValue.S("The Perks of Being a Wallflower"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun putRecordPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
 'info' : ?}"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2020"))
    parameters.add(AttributeValue.S("My Movie"))
    parameters.add(AttributeValue.S("No Info"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Added new movie.")
}

suspend fun updateTableItemPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian C.
 Cooper\",\"Ernest B. Schoedsack\' where year=? and title=?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    parameters.add(AttributeValue.S("The East"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Item was updated!")
}

// Query the table where the year is 2013.
suspend fun queryTablePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year = ?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}
```

```
suspend fun deleteTablePartiQL(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun executeStatementPartiQL(
    ddb: DynamoDbClient,
    statementVal: String,
    parametersVal: List<AttributeValue>,
): ExecuteStatementResponse {
    val request =
        ExecuteStatementRequest {
            statement = statementVal
            parameters = parametersVal
        }

    return ddb.executeStatement(request)
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ExecuteStatement를 참조하십시오.

SDK for Kotlin을 사용한 Amazon EC2 예제

다음 코드 예제에서는 Amazon EC2와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시작

Hello Amazon EC2

다음 코드 예제에서는 Amazon EC2 사용을 시작하는 방법을 보여줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->

        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
 ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeSecurityGroups를 참조하십시오.

주제

- 기본 사항
- 작업

기본 사항

기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 키 페어 및 보안 그룹을 생성합니다.
- Amazon Machine Image(AMI) 및 호환되는 인스턴스 유형을 선택한 다음 인스턴스를 생성합니다.
- 인스턴스를 중지한 후 다시 시작합니다.
- 인스턴스와 탄력적 IP 주소 연결.
- SSH로 인스턴스에 연결한 다음 리소스를 정리합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin example performs the following tasks:

 1. Creates an RSA key pair and saves the private key data as a .pem file.
 2. Lists key pairs.
 3. Creates a security group for the default VPC.
 4. Displays security group information.
 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 6. Gets more information about the image.
 7. Gets a list of instance types that are compatible with the selected AMI's
 architecture.
 8. Creates an instance with the key pair, security group, AMI, and an instance
 type.
```

```
 9. Displays information about the instance.
 10. Stops the instance and waits for it to stop.
 11. Starts the instance and waits for it to start.
 12. Allocates an Elastic IP address and associates it with the instance.
 13. Displays SSH connection info for the instance.
 14. Disassociates and deletes the Elastic IP address.
 15. Terminates the instance.
 16. Deletes the security group.
 17. Deletes the key pair.
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <keyName> <fileName> <groupName> <groupDesc> <vpcId> <myIpAddress>

        Where:
            keyName - A key pair name (for example, TestKeyPair).
            fileName - A file name where the key information is written to.
            groupName - The name of the security group.
            groupDesc - The description of the security group.
            vpcId - A VPC ID. You can get this value from the AWS Management
 Console.
            myIpAddress - The IP address of your development machine.

"""

    if (args.size != 6) {
        println(usage)
        exitProcess(0)
    }

    val keyName = args[0]
    val fileName = args[1]
    val groupName = args[2]
    val groupDesc = args[3]
    val vpcId = args[4]
    val myIpAddress = args[5]
    var newInstanceId: String? = ""

    println(DASHES)
    println("Welcome to the Amazon EC2 example scenario.")
```

```
    println(DASHES)

    println(DASHES)
    println("1. Create an RSA key pair and save the private key material as a .pem
file.")
    createKeyPairSc(keyName, fileName)
    println(DASHES)

    println(DASHES)
    println("2. List key pairs.")
    describeEC2KeysSc()
    println(DASHES)

    println(DASHES)
    println("3. Create a security group.")
    val groupId = createEC2SecurityGroupSc(groupName, groupDesc, vpcId, myIpAddress)
    println(DASHES)

    println(DASHES)
    println("4. Display security group info for the newly created security group.")
    describeSecurityGroupsSc(groupId.toString())
    println(DASHES)

    println(DASHES)
    println("5. Get a list of Amazon Linux 2 AMIs and select one with amzn2 in the
name.")
    val instanceId = getParaValuesSc()
    if (instanceId == "") {
        println("The instance Id value isn't valid.")
        exitProcess(0)
    }
    println("The instance Id is $instanceId.")
    println(DASHES)

    println(DASHES)
    println("6. Get more information about an amzn2 image and return the AMI
value.")
    val amiValue = instanceId?.let { describeImageSc(it) }
    if (instanceId == "") {
        println("The instance Id value is invalid.")
        exitProcess(0)
    }
    println("The AMI value is $amiValue.")
    println(DASHES)
```

```
    println(DASHES)
    println("7. Get a list of instance types.")
    val instanceType = getInstanceTypesSc()
    println(DASHES)

    println(DASHES)
    println("8. Create an instance.")
    if (amiValue != null) {
        newInstanceId = runInstanceSc(instanceType, keyName, groupName, amiValue)
        println("The instance Id is $newInstanceId")
    }
    println(DASHES)

    println(DASHES)
    println("9. Display information about the running instance. ")
    var ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("10.  Stop the instance.")
    if (newInstanceId != null) {
        stopInstanceSc(newInstanceId)
    }
    println(DASHES)

    println(DASHES)
    println("11.  Start the instance.")
    if (newInstanceId != null) {
        startInstanceSc(newInstanceId)
    }
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("12. Allocate an Elastic IP address and associate it with the
instance.")
    val allocationId = allocateAddressSc()
    println("The allocation Id value is $allocationId")
    val associationId = associateAddressSc(newInstanceId, allocationId)
```

```
    println("The associate Id value is $associationId")
    println(DASHES)

    println(DASHES)
    println("13. Describe the instance again.")
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("14. Disassociate and release the Elastic IP address.")
    disassociateAddressSc(associationId)
    releaseEC2AddressSc(allocationId)
    println(DASHES)

    println(DASHES)
    println("15. Terminate the instance and use a waiter.")
    if (newInstanceId != null) {
        terminateEC2Sc(newInstanceId)
    }
    println(DASHES)

    println(DASHES)
    println("16. Delete the security group.")
    if (groupId != null) {
        deleteEC2SecGroupSc(groupId)
    }
    println(DASHES)

    println(DASHES)
    println("17. Delete the key pair.")
    deleteKeysSc(keyName)
    println(DASHES)

    println(DASHES)
    println("You successfully completed the Amazon EC2 scenario.")
    println(DASHES)
}

suspend fun deleteKeysSc(keyPair: String) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
```

```
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}

suspend fun deleteEC2SecGroupSc(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted security group with Id $groupIdVal")
    }
}

suspend fun terminateEC2Sc(instanceIdVal: String) {
    val ti =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceIdVal)
        }
    println("Wait for the instance to terminate. This will take a few minutes.")
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.terminateInstances(ti)
        ec2.waitUntilInstanceTerminated {
            // suspend call
            instanceIds = listOf(instanceIdVal)
        }
        println("$instanceIdVal is terminated!")
    }
}

suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
```

```
}

suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}

suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}

suspend fun allocateAddressSc(): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        return allocateResponse.allocationId
    }
}

suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
```

```
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
 minutes.")
        ec2.waitUntilInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}

suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
 minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}

suspend fun describeEC2InstancesSc(newInstanceId: String?): String {
    var pubAddress = ""
    var isRunning = false
    val request =
        DescribeInstancesRequest {
            instanceIds = listOf(newInstanceId.toString())
        }

    while (!isRunning) {
        Ec2Client { region = "us-west-2" }.use { ec2 ->
            val response = ec2.describeInstances(request)
            val state =
```

```
                response.reservations
                    ?.get(0)
                    ?.instances
                    ?.get(0)
                    ?.state
                    ?.name
                    ?. value
            if (state != null) {
                if (state.compareTo("running") == 0) {
                    println("Image id is
 ${response.reservations!!.get(0).instances?.get(0)?.imageId}")
                    println("Instance type is
 ${response.reservations!!.get(0).instances?.get(0)?.instanceType}")
                    println("Instance state is
 ${response.reservations!!.get(0).instances?.get(0)?.state}")
                    pubAddress =
                        response.reservations!!
                            .get(0)
                            .instances
                            ?.get(0)
                            ?.publicIpAddress
                            .toString()
                    println("Instance address is $pubAddress")
                    isRunning = true
                }
            }
        }
    }
    return pubAddress
}

suspend fun runInstanceSc(
    instanceTypeVal: String,
    keyNameVal: String,
    groupNameVal: String,
    amiIdVal: String,
): String {
    val runRequest =
        RunInstancesRequest {
            instanceType = InstanceType.fromValue(instanceTypeVal)
            keyName = keyNameVal
            securityGroups = listOf(groupNameVal)
            maxCount = 1
            minCount = 1
```

```
            imageId = amiIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(runRequest)
        val instanceId = response.instances?.get(0)?.instanceId
        println("Successfully started EC2 Instance $instanceId based on AMI
 $amiIdVal")
        return instanceId.toString()
    }
}

// Get a list of instance types.
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
 ${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
 ${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}

// Display the Description field that corresponds to the instance Id value.
suspend fun describeImageSc(instanceId: String): String? {
    val imagesRequest =
```

```
        DescribeImagesRequest {
            imageIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeImages(imagesRequest)
        println("The description of the first image is
 ${response.images?.get(0)?.description}")
        println("The name of the first image is  ${response.images?.get(0)?.name}")

        // Return the image Id value.
        return response.images?.get(0)?.imageId
    }
}

// Get the Id value of an instance with amzn2 in the name.
suspend fun getParaValuesSc(): String? {
    val parameterRequest =
        GetParametersByPathRequest {
            path = "/aws/service/ami-amazon-linux-latest"
        }

    SsmClient { region = "us-west-2" }.use { ssmClient ->
        val response = ssmClient.getParametersByPath(parameterRequest)
        response.parameters?.forEach { para ->
            println("The name of the para is: ${para.name}")
            println("The type of the para is: ${para.type}")
            println("")
            if (para.name?.let { filterName(it) } == true) {
                return para.value
            }
        }
    }
    return ""
}

fun filterName(name: String): Boolean {
    val parts = name.split("/").toTypedArray()
    val myValue = parts[4]
    return myValue.contains("amzn2")
}

suspend fun describeSecurityGroupsSc(groupId: String) {
    val request =
```

```
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeSecurityGroups(request)
        for (group in response.securityGroups!!) {
            println("Found Security Group with id " + group.groupId.toString() + "
 and group VPC " + group.vpcId)
        }
    }
}

suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
```

```
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group $groupNameVal")
        return resp.groupId
    }
}

suspend fun describeEC2KeysSc() {
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
 ${ keyPair.keyFingerprint}")
        }
    }
}

suspend fun createKeyPairSc(
    keyNameVal: String,
    fileNameVal: String,
) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        val content = response.keyMaterial
        if (content != null) {
            File(fileNameVal).writeText(content)
        }
        println("Successfully created key pair named $keyNameVal")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

작업

`AllocateAddress`

다음 코드 예시에서는 `AllocateAddress`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getAllocateAddress(instanceIdVal: String?): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        val allocationIdVal = allocateResponse.allocationId

        val request =
            AssociateAddressRequest {
                instanceId = instanceIdVal
                allocationId = allocationIdVal
            }

        val associateResponse = ec2.associateAddress(request)
        return associateResponse.associationId
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 AllocateAddress를 참조하십시오.

`AssociateAddress`

다음 코드 예시에서는 `AssociateAddress`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 AssociateAddress를 참조하십시오.

`AuthorizeSecurityGroupIngress`

다음 코드 예시에서는 `AuthorizeSecurityGroupIngress`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createEC2SecurityGroupSc(
```

```
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group $groupNameVal")
        return resp.groupId
    }
```

```
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 AuthorizeSecurityGroupIngress를 참조하십시오.

`CreateKeyPair`

다음 코드 예시에서는 `CreateKeyPair`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createEC2KeyPair(keyNameVal: String) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        println("The key ID is ${response.keyPairId}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateKeyPair를 참조하십시오.

`CreateSecurityGroup`

다음 코드 예시에서는 `CreateSecurityGroup`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createEC2SecurityGroup(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "0.0.0.0/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }
```

```
        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group $groupNameVal")
        return resp.groupId
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateSecurityGroup를 참조하십시오.

`DeleteKeyPair`

다음 코드 예시에서는 `DeleteKeyPair`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteKeys(keyPair: String?) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteKeyPair를 참조하십시오.

`DeleteSecurityGroup`

다음 코드 예시에서는 `DeleteSecurityGroup`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteEC2SecGroup(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted Security Group with id $groupIdVal")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteSecurityGroup를 참조하십시오.

`DescribeInstanceTypes`

다음 코드 예시에서는 `DescribeInstanceTypes`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Get a list of instance types.
```

```
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
 ${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
 ${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeInstanceTypes를 참조하십시오.

`DescribeInstances`

다음 코드 예시에서는 `DescribeInstances`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeEC2Instances() {
    val request =
        DescribeInstancesRequest {
            maxResults = 6
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstances(request)
        response.reservations?.forEach { reservation ->
            reservation.instances?.forEach { instance ->
                println("Instance Id is ${instance.instanceId}")
                println("Image id is ${instance.imageId}")
                println("Instance type is ${instance.instanceType}")
                println("Instance state name is ${instance.state?.name}")
                println("monitoring information is ${instance.monitoring?.state}")
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeInstances를 참조하십시오.

`DescribeKeyPairs`

다음 코드 예시에서는 `DescribeKeyPairs`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeEC2Keys() {
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
 ${ keyPair.keyFingerprint}")
```

```
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeKeyPairs를 참조하십시오.

`DescribeSecurityGroups`

다음 코드 예시에서는 `DescribeSecurityGroups`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->

        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
 ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeSecurityGroups를 참조하십시오.

`DisassociateAddress`

다음 코드 예시에서는 `DisassociateAddress`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DisassociateAddress를 참조하십시오.

`ReleaseAddress`

다음 코드 예시에서는 `ReleaseAddress`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }
```

```
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ReleaseAddress를 참조하십시오.

`RunInstances`

다음 코드 예시에서는 `RunInstances`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createEC2Instance(
    name: String,
    amiId: String,
): String? {
    val request =
        RunInstancesRequest {
            imageId = amiId
            instanceType = InstanceType.T1Micro
            maxCount = 1
            minCount = 1
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(request)
        val instanceId = response.instances?.get(0)?.instanceId
        val tag =
            Tag {
                key = "Name"
                value = name
            }
```

```
        val requestTags =
            CreateTagsRequest {
                resources = listOf(instanceId.toString())
                tags = listOf(tag)
            }
        ec2.createTags(requestTags)
        println("Successfully started EC2 Instance $instanceId based on AMI $amiId")
        return instanceId
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 RunInstances를 참조하십시오.

`StartInstances`

다음 코드 예시에서는 `StartInstances`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
 minutes.")
        ec2.waitUntilInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
```

```
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 StartInstances를 참조하십시오.

`StopInstances`

다음 코드 예시에서는 `StopInstances`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
 minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 StopInstances를 참조하십시오.

`TerminateInstances`

다음 코드 예시에서는 `TerminateInstances`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun terminateEC2(instanceID: String) {
    val request =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceID)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.terminateInstances(request)
        response.terminatingInstances?.forEach { instance ->
            println("The ID of the terminated instance is ${instance.instanceId}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 TerminateInstances를 참조하십시오.

SDK for Kotlin을 사용한 Amazon ECR 예제

다음 코드 예제에서는 Amazon ECR과 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시작

Hello Amazon ECR

다음 코드 예제에서는 Amazon ECR을 사용하여 시작하는 방법을 보여줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.ListImagesRequest
import kotlin.system.exitProcess

suspend fun main(args: Array<String>) {
    val usage = """
            Usage: <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.

    """.trimIndent()

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    val repoName = args[0]
    listImageTags(repoName)
}

suspend fun listImageTags(repoName: String?) {
    val listImages =
        ListImagesRequest {
            repositoryName = repoName
        }

    EcrClient { region = "us-east-1" }.use { ecrClient ->
        val imageResponse = ecrClient.listImages(listImages)
        imageResponse.imageIds?.forEach { imageId ->
```

```
            println("Image tag: ${imageId.imageTag}")
        }
    }
}
```

- API 세부 정보는AWS SDK for Kotlin API 참조의 listImages를 참조하세요.

주제

- 기본 사항
- 작업

기본 사항

기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Amazon ECR 리포지토리를 생성합니다.
- 리포지토리 정책을 설정합니다.
- 리포지토리 URI를 검색합니다.
- Amazon ECR 인증 토큰을 가져옵니다.
- Amazon ECR 리포지토리의 수명 주기 정책을 설정합니다.
- Amazon ECR 리포지토리에 Docker 이미지를 푸시합니다.
- Amazon ECR 리포지토리에 이미지가 있는지 확인합니다.
- 계정의 Amazon ECR 리포지토리를 나열하고 관련 세부 정보를 가져옵니다.
- Amazon ECR 리포지토리를 삭제합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon ECR 기능을 시연하는 대화형 시나리오를 실행합니다.

```
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:
 *
 * /getting_started_scenarios/ecr_scenario/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 = "arn:aws:iam::814548047983:role/Admin"
    var localImageName: String
    var accountId = "814548047983"
    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 메서드의 래퍼 클래스입니다.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.CreateRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DeleteRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DescribeImagesRequest
import aws.sdk.kotlin.services.ecr.model.DescribeRepositoriesRequest
import aws.sdk.kotlin.services.ecr.model.EcrException
import aws.sdk.kotlin.services.ecr.model.GetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.ImageIdentifier
import aws.sdk.kotlin.services.ecr.model.RepositoryAlreadyExistsException
import aws.sdk.kotlin.services.ecr.model.SetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.StartLifecyclePolicyPreviewRequest
```

```
import com.github.dockerjava.api.DockerClient
import com.github.dockerjava.api.command.DockerCmdExecFactory
import com.github.dockerjava.api.model.AuthConfig
import com.github.dockerjava.core.DockerClientBuilder
import com.github.dockerjava.netty.NettyDockerCmdExecFactory
import java.io.IOException
import java.util.Base64

class ECRActions {
    private var dockerClient: DockerClient? = null

    private fun getDockerClient(): DockerClient? {
        val osName = System.getProperty("os.name")
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            val dockerHost = "tcp://localhost:2375" // Use the Docker Desktop
 default port.
            val dockerCmdExecFactory: DockerCmdExecFactory =

 NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000)
            dockerClient =
 DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactory).
        } else {
            dockerClient = DockerClientBuilder.getInstance().build()
        }
        return dockerClient
    }


    /**
     * Sets the lifecycle policy for the specified repository.
     *
     * @param repoName the name of the repository for which to set the lifecycle
 policy.
     */
    suspend fun setLifeCyclePolicy(repoName: String): String? {
        val polText =
            """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
```

```
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14
                    },
                    "action": {
                        "type": "expire"
                    }
                }
            ]
            }

            """.trimIndent()
        val lifecyclePolicyPreviewRequest =
            StartLifecyclePolicyPreviewRequest {
                lifecyclePolicyText = polText
                repositoryName = repoName
            }

        // Execute the request asynchronously.
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest)
            return response.lifecyclePolicyText
        }
    }


    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
```

```
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }


    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }


    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
```

```
            }
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }


    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole the IAM role to be granted access to the repository.
     */
    suspend fun setRepoPolicy(
        repoName: String?,
        iamRole: String?,
    ) {
        val policyDocumentTemplate =
            """
             {
              "Version" : "2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
```

```
            }
        }
    }


    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an empty
string if the operation failed.
     * @throws RepositoryAlreadyExistsException if the repository exists.
     * @throws EcrException         if an error occurs while creating the
repository.
     */
    suspend fun createECRRepository(repoName: String?): String? {
        val request =
            CreateRepositoryRequest {
                repositoryName = repoName
            }

        return try {
            EcrClient { region = "us-east-1" }.use { ecrClient ->
                val response = ecrClient.createRepository(request)
                response.repository?.repositoryArn
            }
        } catch (e: RepositoryAlreadyExistsException) {
            println("Repository already exists: $repoName")
            repoName?.let { getRepoARN(it) }
        } catch (e: EcrException) {
            println("An error occurred: ${e.message}")
            null
        }
    }

    suspend fun getRepoARN(repoName: String): String? {
        // Fetch the existing repository's ARN.
        val describeRequest =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeResponse = ecrClient.describeRepositories(describeRequest)
            return describeResponse.repositories?.get(0)?.repositoryArn
```

```
        }
    }

    fun listLocalImages(): Boolean = try {
        val images = getDockerClient()?.listImagesCmd()?.exec()
        images?.any { image ->
            image.repoTags?.any { tag -> tag.startsWith("echo-text") } ?: false
        } ?: false
    } catch (ex: Exception) {
        println("ERROR: ${ex.message}")
        false
    }


    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    suspend fun pushDockerImage(
        repoName: String,
        imageName: String,
    ) {
        println("Pushing $imageName to $repoName will take a few seconds")
        val authConfig = getAuthConfig(repoName)

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val desRequest =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val describeRepoResponse = ecrClient.describeRepositories(desRequest)
            val repoData =
                describeRepoResponse.repositories?.firstOrNull { it.repositoryName
== repoName }
                    ?: throw RuntimeException("Repository not found: $repoName")

            val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
            if (tagImageCmd != null) {
                tagImageCmd.exec()
```

```
            }
            val pushImageCmd =
                repoData.repositoryUri?.let {
                    dockerClient?.pushImageCmd(it)
                        // ?.withTag("latest")
                        ?.withAuthConfig(authConfig)
                }

            try {
                if (pushImageCmd != null) {
                    pushImageCmd.start().awaitCompletion()
                }
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }


    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
```

```
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }


    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
 empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }

    // Return an AuthConfig.
    private suspend fun getAuthConfig(repoName: String): AuthConfig {
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
```

```
            val decodedToken = String(Base64.getDecoder().decode(token))
            val password = decodedToken.substring(4)

            val request =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val descrRepoResponse = ecrClient.describeRepositories(request)
            val repoData = descrRepoResponse.repositories?.firstOrNull
 { it.repositoryName == repoName }
            val registryURL: String = repoData?.repositoryUri?.split("/")?.get(0) ?:
 ""

            return AuthConfig()
                .withUsername("AWS")
                .withPassword(password)
                .withRegistryAddress(registryURL)
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하세요.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

작업

`CreateRepository`

다음 코드 예시에서는 `CreateRepository`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an empty
string if the operation failed.
     * @throws RepositoryAlreadyExistsException if the repository exists.
     * @throws EcrException         if an error occurs while creating the
repository.
     */
    suspend fun createECRRepository(repoName: String?): String? {
        val request =
            CreateRepositoryRequest {
                repositoryName = repoName
            }

        return try {
            EcrClient { region = "us-east-1" }.use { ecrClient ->
                val response = ecrClient.createRepository(request)
                response.repository?.repositoryArn
            }
        } catch (e: RepositoryAlreadyExistsException) {
            println("Repository already exists: $repoName")
            repoName?.let { getRepoARN(it) }
        } catch (e: EcrException) {
            println("An error occurred: ${e.message}")
            null
        }
    }
```

- API 세부 정보는AWS SDK for Kotlin API 참조의 CreateRepository를 참조하세요.

`DeleteRepository`

다음 코드 예시에서는 `DeleteRepository`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }
```

- API 세부 정보는AWS SDK for Kotlin API 참조의 DeleteRepository를 참조하세요.

`DescribeImages`

다음 코드 예시에서는 `DescribeImages`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
 be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
 be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
```

```
            }
        }
    }
```

- API 세부 정보는AWS SDK for Kotlin API 참조의 DescribeImages를 참조하세요.

`DescribeRepositories`

다음 코드 예시에서는 `DescribeRepositories`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
```

```
                println("No repositories found for the given name.")
                return ""
            }
        }
    }
```

- API 세부 정보는AWS SDK for Kotlin API 참조의 DescribeRepositories를 참조하세요.

`GetAuthorizationToken`

다음 코드 예시에서는 `GetAuthorizationToken`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }
```

- API 세부 정보는AWS SDK for Kotlin API 참조의 GetAuthorizationToken을 참조하세요.

GetRepositoryPolicy

다음 코드 예시에서는 GetRepositoryPolicy을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }
```

- API 세부 정보는AWS SDK for Kotlin API 참조의 GetRepositoryPolicy를 참조하세요.

PushImageCmd

다음 코드 예시에서는 PushImageCmd을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    suspend fun pushDockerImage(
        repoName: String,
        imageName: String,
    ) {
        println("Pushing $imageName to $repoName will take a few seconds")
        val authConfig = getAuthConfig(repoName)

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val desRequest =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val describeRepoResponse = ecrClient.describeRepositories(desRequest)
            val repoData =
                describeRepoResponse.repositories?.firstOrNull { it.repositoryName
== repoName }
                    ?: throw RuntimeException("Repository not found: $repoName")

            val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
            if (tagImageCmd != null) {
                tagImageCmd.exec()
            }
            val pushImageCmd =
                repoData.repositoryUri?.let {
```

```
                dockerClient?.pushImageCmd(it)
                    // ?.withTag("latest")
                    ?.withAuthConfig(authConfig)
            }

        try {
            if (pushImageCmd != null) {
                pushImageCmd.start().awaitCompletion()
            }
            println("The $imageName was pushed to Amazon ECR")
        } catch (e: IOException) {
            throw RuntimeException(e)
        }
    }
}
```

- API 세부 정보는AWS SDK for Kotlin API 참조의 PushImageCmd를 참조하세요.

`SetRepositoryPolicy`

다음 코드 예시에서는 `SetRepositoryPolicy`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Sets the repository policy for the specified ECR repository.
 *
 * @param repoName the name of the ECR repository.
 * @param iamRole the IAM role to be granted access to the repository.
 */
suspend fun setRepoPolicy(
    repoName: String?,
    iamRole: String?,
) {
```

```
        val policyDocumentTemplate =
            """
             {
              "Version" : "2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }
```

- API 세부 정보는AWS SDK for Kotlin API 참조의 SetRepositoryPolicy를 참조하세요.

`StartLifecyclePolicyPreview`

다음 코드 예시에서는 `StartLifecyclePolicyPreview`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
 be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
 be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

- API 세부 정보는AWS SDK for Kotlin API 참조의 StartLifecyclePolicyPreview를 참조하세요.

SDK for Kotlin을 사용한 OpenSearch 서비스 예제

다음 코드 예제에서는 OpenSearch Service와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`CreateDomain`

다음 코드 예시에서는 `CreateDomain`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createNewDomain(domainNameVal: String?) {
    val clusterConfigOb =
        ClusterConfig {
            dedicatedMasterEnabled = true
            dedicatedMasterCount = 3
            dedicatedMasterType =
 OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceType =
 OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceCount = 5
        }
```

```
    val ebsOptionsOb =
        EbsOptions {
            ebsEnabled = true
            volumeSize = 10
            volumeType = VolumeType.Gp2
        }

    val encryptionOptionsOb =
        NodeToNodeEncryptionOptions {
            enabled = true
        }

    val request =
        CreateDomainRequest {
            domainName = domainNameVal
            engineVersion = "OpenSearch_1.0"
            clusterConfig = clusterConfigOb
            ebsOptions = ebsOptionsOb
            nodeToNodeEncryptionOptions = encryptionOptionsOb
        }

    println("Sending domain creation request...")
    OpenSearchClient { region = "us-east-1" }.use { searchClient ->
        val createResponse = searchClient.createDomain(request)
        println("Domain status is ${createResponse.domainStatus}")
        println("Domain Id is ${createResponse.domainStatus?.domainId}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateDomain을 참조하십시오.

`DeleteDomain`

다음 코드 예시에서는 `DeleteDomain`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteSpecificDomain(domainNameVal: String) {
    val request =
        DeleteDomainRequest {
            domainName = domainNameVal
        }
    OpenSearchClient { region = "us-east-1" }.use { searchClient ->
        searchClient.deleteDomain(request)
        println("$domainNameVal was successfully deleted.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteDomain을 참조하십시오.

`ListDomainNames`

다음 코드 예시에서는 `ListDomainNames`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listAllDomains() {
    OpenSearchClient { region = "us-east-1" }.use { searchClient ->
        val response: ListDomainNamesResponse =
 searchClient.listDomainNames(ListDomainNamesRequest {})
        response.domainNames?.forEach { domain ->
            println("Domain name is " + domain.domainName)
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListDomainNames를 참조하십시오.

`UpdateDomainConfig`

다음 코드 예시에서는 `UpdateDomainConfig`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun updateSpecificDomain(domainNameVal: String?) {
    val clusterConfigOb =
        ClusterConfig {
            instanceCount = 3
        }

    val request =
        UpdateDomainConfigRequest {
            domainName = domainNameVal
            clusterConfig = clusterConfigOb
        }

    println("Sending domain update request...")
    OpenSearchClient { region = "us-east-1" }.use { searchClient ->
        val updateResponse = searchClient.updateDomainConfig(request)
        println("Domain update response from Amazon OpenSearch Service:")
        println(updateResponse.toString())
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 UpdateDomainConfig를 참조하십시오.

SDK for Kotlin을 사용한 EventBridge 예제

다음 코드 예제에서는 EventBridge와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시작

Hello EventBridge

다음 코드 예제에서는 EventBridge를 사용하여 시작하는 방법을 보여줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import aws.sdk.kotlin.services.eventbridge.EventBridgeClient
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesRequest
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesResponse

suspend fun main() {
    listBusesHello()
}

suspend fun listBusesHello() {
    val request =
        ListEventBusesRequest {
            limit = 10
        }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val response: ListEventBusesResponse = eventBrClient.listEventBuses(request)
        response.eventBuses?.forEach { bus ->
            println("The name of the event bus is ${bus.name}")
            println("The ARN of the event bus is ${bus.arn}")
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Kotlin API reference의 ListEventBuses를 참조하세요.

주제

- 기본 사항
- 작업

기본 사항

기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 규칙을 만들고 여기에 대상을 추가하세요.
- 규칙을 활성화 및 비활성화합니다.
- 규칙과 대상을 나열하고 업데이트합니다.
- 이벤트를 전송하고 리소스를 정리합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/*
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin example performs the following tasks with Amazon EventBridge:

 1. Creates an AWS Identity and Access Management (IAM) role to use with Amazon
 EventBridge.
```

```
 2. Creates an Amazon Simple Storage Service (Amazon S3) bucket with EventBridge
 events enabled.
 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 4. Lists rules on the event bus.
 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and lets the
 user subscribe to it.
 6. Adds a target to the rule that sends an email to the specified topic.
 7. Creates an EventBridge event that sends an email when an Amazon S3 object is
 created.
 8. Lists targets.
 9. Lists the rules for the same target.
 10. Triggers the rule by uploading a file to the S3 bucket.
 11. Disables a specific rule.
 12. Checks and prints the state of the rule.
 13. Adds a transform to the rule to change the text of the email.
 14. Enables a specific rule.
 15. Triggers the updated rule by uploading a file to the S3 bucket.
 16. Updates the rule to a custom rule pattern.
 17. Sends an event to trigger the rule.
 18. Cleans up resources.
*/
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <roleName> <bucketName> <topicName> <eventRuleName>

    Where:
        roleName - The name of the role to create.
        bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name to
 create.
        topicName - The name of the Amazon Simple Notification Service (Amazon SNS)
 topic to create.
        eventRuleName - The Amazon EventBridge rule name to create.
    """
    val polJSON =
        "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
            "\"Effect\": \"Allow\"," +
            "\"Principal\": {" +
            "\"Service\": \"events.amazonaws.com\"" +
            "}," +
```

```
            "\"Action\": \"sts:AssumeRole\"" +
            "}]" +
            "}"

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val sc = Scanner(System.`in`)
    val roleName = args[0]
    val bucketName = args[1]
    val topicName = args[2]
    val eventRuleName = args[3]

    println(DASHES)
    println("Welcome to the Amazon EventBridge example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS Identity and Access Management (IAM) role to use with
Amazon EventBridge.")
    val roleArn = createIAMRole(roleName, polJSON)
    println(DASHES)

    println(DASHES)
    println("2. Create an S3 bucket with EventBridge events enabled.")
    if (checkBucket(bucketName)) {
        println("$bucketName already exists. Ending this scenario.")
        exitProcess(1)
    }

    createBucket(bucketName)
    delay(3000)
    setBucketNotification(bucketName)
    println(DASHES)

    println(DASHES)
    println("3. Create a rule that triggers when an object is uploaded to Amazon
S3.")
    delay(10000)
    addEventRule(roleArn, bucketName, eventRuleName)
    println(DASHES)
```

```
    println(DASHES)
    println("4. List rules on the event bus.")
    listRules()
    println(DASHES)

    println(DASHES)
    println("5. Create a new SNS topic for testing and let the user subscribe to the
topic.")
    val topicArn = createSnsTopic(topicName)
    println(DASHES)

    println(DASHES)
    println("6. Add a target to the rule that sends an email to the specified
topic.")
    println("Enter your email to subscribe to the Amazon SNS topic:")
    val email = sc.nextLine()
    subEmail(topicArn, email)
    println("Use the link in the email you received to confirm your subscription.
Then press Enter to continue.")
    sc.nextLine()
    println(DASHES)

    println(DASHES)
    println("7. Create an EventBridge event that sends an email when an Amazon S3
object is created.")
    addSnsEventRule(eventRuleName, topicArn, topicName, eventRuleName, bucketName)
    println(DASHES)

    println(DASHES)
    println("8. List targets.")
    listTargets(eventRuleName)
    println(DASHES)

    println(DASHES)
    println(" 9. List the rules for the same target.")
    listTargetRules(topicArn)
    println(DASHES)

    println(DASHES)
    println("10. Trigger the rule by uploading a file to the S3 bucket.")
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)
```

```
    println(DASHES)
    println("11. Disable a specific rule.")
    changeRuleState(eventRuleName, false)
    println(DASHES)

    println(DASHES)
    println("12. Check and print the state of the rule.")
    checkRule(eventRuleName)
    println(DASHES)

    println(DASHES)
    println("13. Add a transform to the rule to change the text of the email.")
    updateSnsEventRule(topicArn, eventRuleName)
    println(DASHES)

    println(DASHES)
    println("14. Enable a specific rule.")
    changeRuleState(eventRuleName, true)
    println(DASHES)

    println(DASHES)
    println("15. Trigger the updated rule by uploading a file to the S3 bucket.")
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
    println("16. Update the rule to a custom rule pattern.")
    updateToCustomRule(eventRuleName)
    println("Updated event rule $eventRuleName to use a custom pattern.")
    updateCustomRuleTargetWithTransform(topicArn, eventRuleName)
    println("Updated event target $topicArn.")
    println(DASHES)

    println(DASHES)
    println("17. Send an event to trigger the rule. This will trigger a subscription
email.")
    triggerCustomRule(email)
    println("Events have been sent. Press Enter to continue.")
    sc.nextLine()
    println(DASHES)
```

```
    println(DASHES)
    println("18. Clean up resources.")
    println("Do you want to clean up resources (y/n)")
    val ans = sc.nextLine()
    if (ans.compareTo("y") == 0) {
        cleanupResources(topicArn, eventRuleName, bucketName, roleName)
    } else {
        println("The resources will not be cleaned up. ")
    }
    println(DASHES)

    println(DASHES)
    println("The Amazon EventBridge example scenario has successfully completed.")
    println(DASHES)
}

suspend fun cleanupResources(
    topicArn: String?,
    eventRuleName: String?,
    bucketName: String?,
    roleName: String?,
) {
    println("Removing all targets from the event rule.")
    deleteTargetsFromRule(eventRuleName)
    deleteRuleByName(eventRuleName)
    deleteSNSTopic(topicArn)
    deleteS3Bucket(bucketName)
    deleteRole(roleName)
}

suspend fun deleteRole(roleNameVal: String?) {
    val policyArnVal = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
    val policyRequest =
        DetachRolePolicyRequest {
            policyArn = policyArnVal
            roleName = roleNameVal
        }
    IamClient { region = "us-east-1" }.use { iam ->
        iam.detachRolePolicy(policyRequest)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")

        // Delete the role.
        val roleRequest =
            DeleteRoleRequest {
```

```
                roleName = roleNameVal
            }

        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteS3Bucket(bucketName: String?) {
    // Remove all the objects from the S3 bucket.
    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }
    S3Client { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val myObjects = res.contents
        val toDelete = mutableListOf<ObjectIdentifier>()

        if (myObjects != null) {
            for (myValue in myObjects) {
                toDelete.add(
                    ObjectIdentifier {
                        key = myValue.key
                    },
                )
            }
        }

        val delOb =
            Delete {
                objects = toDelete
            }

        val dor =
            DeleteObjectsRequest {
                bucket = bucketName
                delete = delOb
            }
        s3Client.deleteObjects(dor)

        // Delete the S3 bucket.
        val deleteBucketRequest =
            DeleteBucketRequest {
```

```
                bucket = bucketName
            }
        s3Client.deleteBucket(deleteBucketRequest)
        println("You have deleted the bucket and the objects")
    }
}

// Delete the SNS topic.
suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println(" $topicArnVal was deleted.")
    }
}

suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}

suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
```

```
            for (myTarget in allTargets) {
                val removeTargetsRequest =
                    RemoveTargetsRequest {
                        rule = eventRuleName
                        ids = listOf(myTarget.id.toString())
                    }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
}

suspend fun triggerCustomRule(email: String) {
    val json =
        "{" +
            "\"UserEmail\": \"" + email + "\"," +
            "\"Message\": \"This event was generated by example code.\"" +
            "\"UtcTime\": \"Now.\"" +
            "}"

    val entry =
        PutEventsRequestEntry {
            source = "ExampleSource"
            detail = json
            detailType = "ExampleType"
        }

    val eventsRequest =
        PutEventsRequest {
            this.entries = listOf(entry)
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putEvents(eventsRequest)
    }
}

suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()
```

```
    val inputTransformerOb =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\""
        }

    val target =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformerOb
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(target)
            eventBusName = null
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun updateToCustomRule(ruleName: String?) {
    val customEventsPattern =
        "{" +
            "\"source\": [\"ExampleSource\"]," +
            "\"detail-type\": [\"ExampleType\"]" +
            "}"
    val request =
        PutRuleRequest {
            name = ruleName
            description = "Custom test rule"
            eventPattern = customEventsPattern
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putRule(request)
    }
}

// Update an Amazon S3 object created rule with a transform on the target.
suspend fun updateSnsEventRule(
```

```
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()
    val myMap = mutableMapOf<String, String>()
    myMap["bucket"] = "$.detail.bucket.name"
    myMap["time"] = "$.time"

    val inputTransOb =
        InputTransformer {
            inputTemplate = "\"Notification: an object was uploaded to bucket
 <bucket> at <time>.\""
            inputPathsMap = myMap
        }
    val targetOb =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransOb
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(targetOb)
            eventBusName = null
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}
```

```
suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}

// Create and upload a file to an S3 bucket to trigger an event.
@Throws(IOException::class)
suspend fun uploadTextFiletoS3(bucketName: String?) {
    val fileSuffix = SimpleDateFormat("yyyyMMddHHmmss").format(Date())
    val fileName = "TextFile$fileSuffix.txt"
    val myFile = File(fileName)
    val fw = FileWriter(myFile.absoluteFile)
    val bw = BufferedWriter(fw)
    bw.write("This is a sample file for testing uploads.")
    bw.close()

    val putOb =
        PutObjectRequest {
            bucket = bucketName
            key = fileName
            body = myFile.asByteStream()
        }
```

```
    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.putObject(putOb)
    }
}

suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}

suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}

// Add a rule that triggers an SNS target when a file is uploaded to an S3 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
```

```
            id = targetID
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request =
        PutTargetsRequest {
            eventBusName = null
            targets = targetsOb
            rule = ruleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target $topicName
 for bucket $bucketName.")
    }
}

suspend fun subEmail(
    topicArnVal: String?,
    email: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" Subscription ARN: ${result.subscriptionArn}")
    }
}

suspend fun createSnsTopic(topicName: String): String? {
    val topicPolicy =
        "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
```

```
            "\"Sid\": \"EventBridgePublishTopic\"," +
            "\"Effect\": \"Allow\"," +
            "\"Principal\": {" +
            "\"Service\": \"events.amazonaws.com\"" +
            "}," +
            "\"Resource\": \"*\"," +
            "\"Action\": \"sns:Publish\"" +
            "}]" +
            "}"

    val topicAttributes = mutableMapOf<String, String>()
    topicAttributes["Policy"] = topicPolicy

    val topicRequest =
        CreateTopicRequest {
            name = topicName
            attributes = topicAttributes
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        println("Added topic $topicName for email subscriptions.")
        return response.topicArn
    }
}

suspend fun listRules() {
    val rulesRequest =
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}

// Create a new event rule that triggers when an Amazon S3 object is created in a
 bucket.
```

```
suspend fun addEventRule(
    roleArnVal: String?,
    bucketName: String,
    eventRuleName: String?,
) {
    val pattern = """{
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
        "bucket": {
            "name": ["$bucketName"]
            }
        }
    }"""

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}

// Set the Amazon S3 bucket notification configuration.
suspend fun setBucketNotification(bucketName: String) {
    val eventBridgeConfig =
        EventBridgeConfiguration {
        }

    val configuration =
        NotificationConfiguration {
            eventBridgeConfiguration = eventBridgeConfig
        }

    val configurationRequest =
        PutBucketNotificationConfigurationRequest {
            bucket = bucketName
            notificationConfiguration = configuration
```

```
            skipDestinationValidation = true
        }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.putBucketNotificationConfiguration(configurationRequest)
        println("Added bucket $bucketName with EventBridge events enabled.")
    }
}

// Create an S3 bucket using a waiter.
suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        s3.waitUntilBucketExists {
            bucket = bucketName
        }
        println("$bucketName is ready")
    }
}

suspend fun checkBucket(bucketName: String?): Boolean {
    try {
        // Determine if the S3 bucket exists.
        val headBucketRequest =
            HeadBucketRequest {
                bucket = bucketName
            }

        S3Client { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            return true
        }
    } catch (e: S3Exception) {
        System.err.println(e.message)
    }
    return false
}

suspend fun createIAMRole(
```

```
    rolenameVal: String?,
    polJSON: String?,
): String? {
    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = polJSON
            description = "Created using the AWS SDK for Kotlin"
        }

    val rolePolicyRequest =
        AttachRolePolicyRequest {
            roleName = rolenameVal
            policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
        }

    IamClient { region = "us-east-1" }.use { iam ->
        val response = iam.createRole(request)
        iam.attachRolePolicy(rolePolicyRequest)
        return response.role?.arn
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API reference의 다음 주제를 참조하세요.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

작업

`DeleteRule`

다음 코드 예시에서는 `DeleteRule`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteRule을 참조하십시오.

`DescribeRule`

다음 코드 예시에서는 `DescribeRule`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeRule을 참조하십시오.

`DisableRule`

다음 코드 예시에서는 `DisableRule`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
```

```
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DisableRule을 참조하십시오.

`EnableRule`

다음 코드 예시에서는 `EnableRule`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
```

```
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 EnableRule를 참조하십시오.

`ListRuleNamesByTarget`

다음 코드 예시에서는 `ListRuleNamesByTarget`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListRuleNamesByTarget을 참조하십시오.

`ListRules`

다음 코드 예시에서는 `ListRules`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listRules() {
    val rulesRequest =
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListRules를 참조하십시오.

`ListTargetsByRule`

다음 코드 예시에서는 `ListTargetsByRule`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListTargetsByRule을 참조하십시오.

`PutEvents`

다음 코드 예시에서는 `PutEvents`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun triggerCustomRule(email: String) {
    val json =
        "{" +
```

```
            "\"UserEmail\": \"" + email + "\"," +
            "\"Message\": \"This event was generated by example code.\"" +
            "\"UtcTime\": \"Now.\"" +
            "}"

    val entry =
        PutEventsRequestEntry {
            source = "ExampleSource"
            detail = json
            detailType = "ExampleType"
        }

    val eventsRequest =
        PutEventsRequest {
            this.entries = listOf(entry)
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putEvents(eventsRequest)
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 PutEvents를 참조하십시오.

`PutRule`

다음 코드 예시에서는 `PutRule`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

예약된 규칙을 생성합니다.

```
suspend fun createScRule(
    ruleName: String?,
    cronExpression: String?,
```

```
) {
    val ruleRequest =
        PutRuleRequest {
            name = ruleName
            eventBusName = "default"
            scheduleExpression = cronExpression
            state = RuleState.Enabled
            description = "A test rule that runs on a schedule created by the Kotlin
 API"
        }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

Amazon Simple Storage Service 버킷에 객체가 추가될 때 트리거되는 규칙을 생성합니다.

```
// Create a new event rule that triggers when an Amazon S3 object is created in a
 bucket.
suspend fun addEventRule(
    roleArnVal: String?,
    bucketName: String,
    eventRuleName: String?,
) {
    val pattern = """{
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
        "bucket": {
            "name": ["$bucketName"]
            }
        }
    }"""

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
```

```
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 PutRule을 참조하십시오.

`PutTargets`

다음 코드 예시에서는 `PutTargets`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Add a rule that triggers an SNS target when a file is uploaded to an S3 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
            id = targetID
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request =
```

```
        PutTargetsRequest {
            eventBusName = null
            targets = targetsOb
            rule = ruleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target $topicName
 for bucket $bucketName.")
    }
}
```

규칙의 대상에 입력 변환기를 추가합니다.

```
suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\""
        }

    val target =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformerOb
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(target)
            eventBusName = null
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
```

```
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 PutTargets를 참조하십시오.

`RemoveTargets`

다음 코드 예시에서는 `RemoveTargets`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest =
                    RemoveTargetsRequest {
                        rule = eventRuleName
                        ids = listOf(myTarget.id.toString())
                    }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
```

```
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 RemoveTargets를 참조하십시오.

AWS Glue SDK for Kotlin을 사용한 예제

다음 코드 예제에서는 Kotlin용 AWS SDK를와 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS Glue.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 전체 소스 코드에 대한 링크가 포함되어 있습니다.

주제

- 기본 사항
- 작업

기본 사항

기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 퍼블릭 Amazon S3 버킷을 크롤링하고 CSV 형식의 메타데이터 데이터베이스를 생성하는 크롤러를 생성합니다.
- 의 데이터베이스 및 테이블에 대한 정보를 나열합니다 AWS Glue Data Catalog.
- 작업을 생성하여 S3 버킷에서 CSV 데이터를 추출하고, 데이터를 변환하며, JSON 형식의 출력을 다른 S3 버킷으로 로드합니다.
- 작업 실행에 대한 정보를 나열하고 변환된 데이터를 확인하며 리소스를 정리합니다.

자세한 내용은 자습서: AWS Glue Studio 시작하기를 참조하세요.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName> <scriptLocation>
 <locationUri>

        Where:
            iam - The Amazon Resource Name (ARN) of the AWS Identity and Access
 Management (IAM) role that has AWS Glue and Amazon Simple Storage Service (Amazon
 S3) permissions.
            s3Path - The Amazon Simple Storage Service (Amazon S3) target that
 contains data (for example, CSV data).
            cron - A cron expression used to specify the schedule (for example,
 cron(15 12 * * ? *).
            dbName - The database name.
            crawlerName - The name of the crawler.
            jobName - The name you assign to this job definition.
            scriptLocation - Specifies the Amazon S3 path to a script that runs a
 job.
            locationUri - Specifies the location of the database
        """

    if (args.size != 8) {
        println(usage)
        exitProcess(1)
    }

    val iam = args[0]
    val s3Path = args[1]
    val cron = args[2]
    val dbName = args[3]
    val crawlerName = args[4]
    val jobName = args[5]
    val scriptLocation = args[6]
```

```
    val locationUri = args[7]

    println("About to start the AWS Glue Scenario")
    createDatabase(dbName, locationUri)
    createCrawler(iam, s3Path, cron, dbName, crawlerName)
    getCrawler(crawlerName)
    startCrawler(crawlerName)
    getDatabase(dbName)
    getGlueTables(dbName)
    createJob(jobName, iam, scriptLocation)
    startJob(jobName)
    getJobs()
    getJobRuns(jobName)
    deleteJob(jobName)
    println("*** Wait for 5 MIN so the $crawlerName is ready to be deleted")
    TimeUnit.MINUTES.sleep(5)
    deleteMyDatabase(dbName)
    deleteCrawler(crawlerName)
}

suspend fun createDatabase(
    dbName: String?,
    locationUriVal: String?,
) {
    val input =
        DatabaseInput {
            description = "Built with the AWS SDK for Kotlin"
            name = dbName
            locationUri = locationUriVal
        }

    val request =
        CreateDatabaseRequest {
            databaseInput = input
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createDatabase(request)
        println("The database was successfully created")
    }
}

suspend fun createCrawler(
    iam: String?,
```

```
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    val targetList = ArrayList<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val crawlerRequest =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Java API"
            targets = targetOb
            role = iam
            schedule = cron
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createCrawler(crawlerRequest)
        println("$crawlerName was successfully created")
    }
}

suspend fun getCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
```

```
    }
}

suspend fun startCrawler(crawlerName: String) {
    val crawlerRequest =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.startCrawler(crawlerRequest)
        println("$crawlerName was successfully started.")
    }
}

suspend fun getDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}

suspend fun getGlueTables(dbName: String?) {
    val tableRequest =
        GetTablesRequest {
            databaseName = dbName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getTables(tableRequest)
        response.tableList?.forEach { tableName ->
            println("Table name is ${tableName.name}")
        }
    }
}

suspend fun startJob(jobNameVal: String?) {
    val runRequest =
```

```
        StartJobRunRequest {
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.startJobRun(runRequest)
        println("The job run Id is ${response.jobRunId}")
    }
}

suspend fun createJob(
    jobName: String,
    iam: String?,
    scriptLocationVal: String?,
) {
    val commandOb =
        JobCommand {
            pythonVersion = "3"
            name = "MyJob1"
            scriptLocation = scriptLocationVal
        }

    val jobRequest =
        CreateJobRequest {
            description = "A Job created by using the AWS SDK for Java V2"
            glueVersion = "2.0"
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            name = jobName
            role = iam
            command = commandOb
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createJob(jobRequest)
        println("$jobName was successfully created.")
    }
}

suspend fun getJobs() {
    val request =
        GetJobsRequest {
```

```
            maxResults = 10
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobs(request)
        response.jobs?.forEach { job ->
            println("Job name is ${job.name}")
        }
    }
}

suspend fun getJobRuns(jobNameVal: String?) {
    val request =
        GetJobRunsRequest {
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobRuns(request)
        response.jobRuns?.forEach { job ->
            println("Job name is ${job.jobName}")
        }
    }
}

suspend fun deleteJob(jobNameVal: String) {
    val jobRequest =
        DeleteJobRequest {
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteJob(jobRequest)
        println("$jobNameVal was successfully deleted")
    }
}

suspend fun deleteMyDatabase(databaseName: String) {
    val request =
        DeleteDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
```

```
        glueClient.deleteDatabase(request)
        println("$databaseName was successfully deleted")
    }
}

suspend fun deleteCrawler(crawlerName: String) {
    val request =
        DeleteCrawlerRequest {
            name = crawlerName
        }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

작업

`CreateCrawler`

다음 코드 예시에서는 `CreateCrawler`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createGlueCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    // Add the S3Target to a list.
    val targetList = mutableListOf<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val request =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Kotlin API"
            targets = targetOb
            role = iam
```

```
            schedule = cron
        }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.createCrawler(request)
        println("$crawlerName was successfully created")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateCrawler를 참조하십시오.

`GetCrawler`

다음 코드 예시에서는 `GetCrawler`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getSpecificCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetCrawler를 참조하십시오.

`GetDatabase`

다음 코드 예시에서는 `GetDatabase`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getSpecificDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetDatabase를 참조하십시오.

`StartCrawler`

다음 코드 예시에서는 `StartCrawler`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun startSpecificCrawler(crawlerName: String?) {
    val request =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.startCrawler(request)
        println("$crawlerName was successfully started.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 StartCrawler를 참조하십시오.

SDK for Kotlin을 사용한 IAM 예제

다음 코드 예제에서는 AWS SDK for Kotlin을 IAM과 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 전체 소스 코드에 대한 링크가 포함되어 있습니다.

주제

- 기본 사항
- 작업

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 사용자를 생성하고 역할을 수임하는 방법을 보여줍니다.

> **Warning**
>
> 보안 위험을 방지하려면 목적별 소프트웨어를 개발하거나 실제 데이터로 작업할 때 IAM 사용자를 인증에 사용하지 마세요. 대신 AWS IAM Identity Center과 같은 보안 인증 공급자를 통한 페더레이션을 사용하십시오.

- 권한이 없는 사용자를 생성합니다.
- 계정에 대한 Amazon S3 버킷을 나열할 수 있는 권한을 부여하는 역할을 생성합니다.
- 사용자가 역할을 수임할 수 있도록 정책을 추가합니다.
- 역할을 수임하고 임시 보안 인증 정보를 사용하여 S3 버킷을 나열한 후 리소스를 정리합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

IAM 사용자 작업을 래핑하는 함수를 생성합니다.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <username> <policyName> <roleName> <roleSessionName> <fileLocation>
 <bucketName>

    Where:
        username - The name of the IAM user to create.
        policyName - The name of the policy to create.
        roleName - The name of the role to create.
        roleSessionName - The name of the session required for the assumeRole
 operation.
        fileLocation - The file location to the JSON required to create the role
 (see Readme).
        bucketName - The name of the Amazon S3 bucket from which objects are read.
    """
```

```
    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val userName = args[0]
    val policyName = args[1]
    val roleName = args[2]
    val roleSessionName = args[3]
    val fileLocation = args[4]
    val bucketName = args[5]

    createUser(userName)
    println("$userName was successfully created.")

    val polArn = createPolicy(policyName)
    println("The policy $polArn was successfully created.")

    val roleArn = createRole(roleName, fileLocation)
    println("$roleArn was successfully created.")
    attachRolePolicy(roleName, polArn)

    println("*** Wait for 1 MIN so the resource is available.")
    delay(60000)
    assumeGivenRole(roleArn, roleSessionName, bucketName)

    println("*** Getting ready to delete the AWS resources.")
    deleteRole(roleName, polArn)
    deleteUser(userName)
    println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

```
suspend fun createPolicy(policyNameVal: String?): String {
    val policyDocumentValue: String =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}"

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(
    rolenameVal: String?,
    fileLocation: String?,
): String? {
    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toJSONString()
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
```

```
}

suspend fun attachRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role $roleNameVal")
    }
}

fun checkMyList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
```

```
        }
    }
    return 0
}

suspend fun assumeGivenRole(
    roleArnVal: String?,
    roleSessionNameVal: String?,
    bucketName: String,
) {
    val stsClient =
        StsClient {
            region = "us-east-1"
        }

    val roleRequest =
        AssumeRoleRequest {
            roleArn = roleArnVal
            roleSessionName = roleSessionNameVal
        }

    val roleResponse = stsClient.assumeRole(roleRequest)
    val myCreds = roleResponse.credentials
    val key = myCreds?.accessKeyId
    val secKey = myCreds?.secretAccessKey
    val secToken = myCreds?.sessionToken

    val staticCredentials =
        StaticCredentialsProvider {
            accessKeyId = key
            secretAccessKey = secKey
            sessionToken = secToken
        }

    // List all objects in an Amazon S3 bucket using the temp creds.
    val s3 =
        S3Client {
            credentialsProvider = staticCredentials
            region = "us-east-1"
        }

    println("Created a S3Client using temp credentials.")
    println("Listing objects in $bucketName")
```

```
    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }

    val response = s3.listObjects(listObjects)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun deleteRole(
    roleNameVal: String,
    polArn: String,
) {
    val iam = IamClient { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest =
        DetachRolePolicyRequest {
            policyArn = polArn
            roleName = roleNameVal
        }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request =
        DeletePolicyRequest {
            policyArn = polArn
        }

    iam.deletePolicy(request)
    println("*** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest =
        DeleteRoleRequest {
            roleName = roleNameVal
        }

    iam.deleteRole(roleRequest)
    println("*** Successfully deleted $roleNameVal")
```

```
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient { region = "AWS_GLOBAL" }
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

작업

`AttachRolePolicy`

다음 코드 예시에서는 `AttachRolePolicy`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun attachIAMRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
```

```
        println("Successfully attached policy $policyArnVal to role $roleNameVal")
    }
}

fun checkList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 AttachRolePolicy를 참조하십시오.

`CreateAccessKey`

다음 코드 예시에서는 `CreateAccessKey`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createIAMAccessKey(user: String?): String {
    val request =
        CreateAccessKeyRequest {
            userName = user
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
```

```
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateAccessKey를 참조하십시오.

`CreateAccountAlias`

다음 코드 예시에서는 `CreateAccountAlias`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createIAMAccountAlias(alias: String) {
    val request =
        CreateAccountAliasRequest {
            accountAlias = alias
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateAccountAlias를 참조하십시오.

`CreatePolicy`

다음 코드 예시에서는 `CreatePolicy`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createIAMPolicy(policyNameVal: String?): String {
    val policyDocumentVal =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"dynamodb:DeleteItem\"," +
            "            \"dynamodb:GetItem\"," +
            "            \"dynamodb:PutItem\"," +
            "            \"dynamodb:Scan\"," +
            "            \"dynamodb:UpdateItem\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}"

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreatePolicy를 참조하십시오.

`CreateUser`

다음 코드 예시에서는 `CreateUser`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createIAMUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateUser를 참조하십시오.

`DeleteAccessKey`

다음 코드 예시에서는 `DeleteAccessKey`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteKey(
```

```
    userNameVal: String,
    accessKey: String,
) {
    val request =
        DeleteAccessKeyRequest {
            accessKeyId = accessKey
            userName = userNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccessKey(request)
        println("Successfully deleted access key $accessKey from $userNameVal")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteAccessKey를 참조하십시오.

`DeleteAccountAlias`

다음 코드 예시에서는 `DeleteAccountAlias`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteIAMAccountAlias(alias: String) {
    val request =
        DeleteAccountAliasRequest {
            accountAlias = alias
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteAccountAlias를 참조하십시오.

`DeletePolicy`

다음 코드 예시에서는 `DeletePolicy`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {
    val request =
        DeletePolicyRequest {
            policyArn = policyARNVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeletePolicy를 참조하십시오.

`DeleteUser`

다음 코드 예시에서는 `DeleteUser`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteIAMUser(userNameVal: String) {
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    // To delete a user, ensure that the user's access keys are deleted first.
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteUser를 참조하십시오.

`DetachRolePolicy`

다음 코드 예시에서는 `DetachRolePolicy`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun detachPolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
```

```
    val request =
        DetachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DetachRolePolicy를 참조하십시오.

`GetPolicy`

다음 코드 예시에서는 `GetPolicy`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getIAMPolicy(policyArnVal: String?) {
    val request =
        GetPolicyRequest {
            policyArn = policyArnVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetPolicy를 참조하십시오.

`ListAccessKeys`

다음 코드 예시에서는 `ListAccessKeys`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listKeys(userNameVal: String?) {
    val request =
        ListAccessKeysRequest {
            userName = userNameVal
        }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListAccessKeys를 참조하십시오.

`ListAccountAliases`

다음 코드 예시에서는 `ListAccountAliases`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listAliases() {
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListAccountAliases를 참조하십시오.

`ListUsers`

다음 코드 예시에서는 `ListUsers`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listAllUsers() {
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listUsers(ListUsersRequest { })
        response.users?.forEach { user ->
            println("Retrieved user ${user.userName}")
            val permissionsBoundary = user.permissionsBoundary
            if (permissionsBoundary != null) {
                println("Permissions boundary details
 ${permissionsBoundary.permissionsBoundaryType}")
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListUsers를 참조하십시오.

`UpdateUser`

다음 코드 예시에서는 `UpdateUser`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun updateIAMUser(
    curName: String?,
    newName: String?,
) {
    val request =
        UpdateUserRequest {
            userName = curName
            newUserName = newName
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.updateUser(request)
        println("Successfully updated user to $newName")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 UpdateUser를 참조하십시오.

AWS IoT SDK for Kotlin을 사용한 예제

다음 코드 예제에서는 Kotlin용 AWS SDK를와 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS IoT.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시작

안녕하세요 AWS IoT

다음 코드 예제에서는 AWS IoT의 사용을 시작하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.ListThingsRequest

suspend fun main() {
    println("A listing of your AWS IoT Things:")
    listAllThings()
}

suspend fun listAllThings() {
    val thingsRequest =
        ListThingsRequest {
            maxResults = 10
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listThings(thingsRequest)
        val thingList = response.things
        if (thingList != null) {
            for (attribute in thingList) {
                println("Thing name ${attribute.thingName}")
                println("Thing ARN: ${attribute.thingArn}")
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 listThings를 참조하세요.

주제

- 기본 사항
- 작업

기본 사항

기본 사항 알아보기

다음 코드 예제는 AWS IoT 디바이스 관리 작업 방법을 보여줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.Action
import aws.sdk.kotlin.services.iot.model.AttachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.AttributePayload
import aws.sdk.kotlin.services.iot.model.CreateThingRequest
import aws.sdk.kotlin.services.iot.model.CreateTopicRuleRequest
import aws.sdk.kotlin.services.iot.model.DeleteCertificateRequest
import aws.sdk.kotlin.services.iot.model.DeleteThingRequest
import aws.sdk.kotlin.services.iot.model.DescribeEndpointRequest
import aws.sdk.kotlin.services.iot.model.DescribeThingRequest
import aws.sdk.kotlin.services.iot.model.DetachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.ListTopicRulesRequest
import aws.sdk.kotlin.services.iot.model.SearchIndexRequest
import aws.sdk.kotlin.services.iot.model.SnsAction
import aws.sdk.kotlin.services.iot.model.TopicRulePayload
import aws.sdk.kotlin.services.iot.model.UpdateThingRequest
import aws.sdk.kotlin.services.iotdataplane.IotDataPlaneClient
import aws.sdk.kotlin.services.iotdataplane.model.GetThingShadowRequest
```

```
import aws.sdk.kotlin.services.iotdataplane.model.UpdateThingShadowRequest
import aws.smithy.kotlin.runtime.content.ByteStream
import aws.smithy.kotlin.runtime.content.toByteArray
import java.util.Scanner
import java.util.regex.Pattern
import kotlin.system.exitProcess

/**
 * Before running this Kotlin code example, ensure that your development environment
 * is set up, including configuring your credentials.
 *
 * For detailed instructions, refer to the following documentation topic:
 * [Setting Up Your Development Environment](https://docs.aws.amazon.com/sdk-for-
kotlin/latest/developer-guide/setup.html)
 *
 * This code example requires an SNS topic and an IAM Role.
 * Follow the steps in the documentation to set up these resources:
 *
 * - [Creating an SNS Topic](https://docs.aws.amazon.com/sns/latest/dg/sns-getting-
started.html#step-create-topic)
 * - [Creating an IAM Role](https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html)
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")
val TOPIC = "your-iot-topic"

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage:
            <roleARN> <snsAction>

        Where:
            roleARN - The ARN of an IAM role that has permission to work with AWS
 IOT.
            snsAction  - An ARN of an SNS topic.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }
```

```
    var thingName: String
    val roleARN = args[0]
    val snsAction = args[1]
    val scanner = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the AWS IoT example scenario.")
    println(
        """
        This example program demonstrates various interactions with the AWS Internet
of Things (IoT) Core service.
        The program guides you through a series of steps, including creating an IoT
thing, generating a device certificate,
        updating the thing with attributes, and so on.

        It utilizes the AWS SDK for Kotlin and incorporates functionality for
creating and managing IoT things, certificates, rules,
        shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
        developers working with AWS IoT in a Kotlin environment.
        """.trimIndent(),
    )

    print("Press Enter to continue...")
    scanner.nextLine()
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS IoT thing.")
    println(
        """
        An AWS IoT thing represents a virtual entity in the AWS IoT service that can
be associated with a physical device.
        """.trimIndent(),
    )
    // Prompt the user for input.
    print("Enter thing name: ")
    thingName = scanner.nextLine()
    createIoTThing(thingName)
    describeThing(thingName)
    println(DASHES)

    println(DASHES)
```

```
    println("2. Generate a device certificate.")
    println(
        """
        A device certificate performs a role in securing the communication between
devices (things) and the AWS IoT platform.
        """.trimIndent(),
    )

    print("Do you want to create a certificate for $thingName? (y/n)")
    val certAns = scanner.nextLine()
    var certificateArn: String? = ""
    if (certAns != null && certAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
        certificateArn = createCertificate()
        println("Attach the certificate to the AWS IoT thing.")
        attachCertificateToThing(thingName, certificateArn)
    } else {
        println("A device certificate was not created.")
    }
    println(DASHES)

    println(DASHES)
    println("3. Update an AWS IoT thing with Attributes.")
    println(
        """
        IoT thing attributes, represented as key-value pairs, offer a pivotal
advantage in facilitating efficient data
        management and retrieval within the AWS IoT ecosystem.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateThing(thingName)
    println(DASHES)

    println(DASHES)
    println("4. Return a unique endpoint specific to the Amazon Web Services
account.")
    println(
        """
        An IoT Endpoint refers to a specific URL or Uniform Resource Locator that
serves as the entry point for communication between IoT devices and the AWS IoT
service.
        """.trimIndent(),
```

```
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    val endpointUrl = describeEndpoint()
    println(DASHES)

    println(DASHES)
    println("5. List your AWS IoT certificates")
    print("Press Enter to continue...")
    scanner.nextLine()
    if (certificateArn!!.isNotEmpty()) {
        listCertificates()
    } else {
        println("You did not create a certificates. Skipping this step.")
    }
    println(DASHES)

    println(DASHES)
    println("6. Create an IoT shadow that refers to a digital representation or
virtual twin of a physical IoT device")
    println(
        """
        A thing shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
        of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between
        the cloud and the device itself. and the AWS IoT service. For example, you
can write and retrieve JSON data from a thing shadow.

        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateShawdowThing(thingName)
    println(DASHES)

    println(DASHES)
    println("7. Write out the state information, in JSON format.")
    print("Press Enter to continue...")
    scanner.nextLine()
    getPayload(thingName)
    println(DASHES)

    println(DASHES)
```

```
    println("8. Creates a rule")
    println(
        """
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
 processed by the rule.
        """.trimIndent(),
    )
    print("Enter Rule name: ")
    val ruleName = scanner.nextLine()
    createIoTRule(roleARN, ruleName, snsAction)
    println(DASHES)

    println(DASHES)
    println("9. List your rules.")
    print("Press Enter to continue...")
    scanner.nextLine()
    listIoTRules()
    println(DASHES)

    println(DASHES)
    println("10. Search things using the name.")
    print("Press Enter to continue...")
    scanner.nextLine()
    val queryString = "thingName:$thingName"
    searchThings(queryString)
    println(DASHES)

    println(DASHES)
    if (certificateArn.length > 0) {
        print("Do you want to detach and delete the certificate for $thingName? (y/
n)")
        val delAns = scanner.nextLine()
        if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
 true)) {
            println("11. You selected to detach amd delete the certificate.")
            print("Press Enter to continue...")
            scanner.nextLine()
            detachThingPrincipal(thingName, certificateArn)
            deleteCertificate(certificateArn)
        } else {
            println("11. You selected not to delete the certificate.")
        }
    } else {
```

```
        println("11. You did not create a certificate so there is nothing to
 delete.")
    }
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS IoT thing.")
    print("Do you want to delete the IoT thing? (y/n)")
    val delAns = scanner.nextLine()
    if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase = true))
 {
        deleteIoTThing(thingName)
    } else {
        println("The IoT thing was not deleted.")
    }
    println(DASHES)

    println(DASHES)
    println("The AWS IoT workflow has successfully completed.")
    println(DASHES)
}

suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}

suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

```
private fun extractCertificateId(certificateArn: String): String? {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
    val arnParts = certificateArn.split(":".toRegex()).dropLastWhile
 { it.isEmpty() }.toTypedArray()
    val certificateIdPart = arnParts[arnParts.size - 1]
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1)
}

suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}

suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}

suspend fun listIoTRules() {
```

```
    val listTopicRulesRequest = ListTopicRulesRequest {}

    IotClient { region = "us-east-1" }.use { iotClient ->
        val listTopicRulesResponse = iotClient.listTopicRules(listTopicRulesRequest)
        println("List of IoT rules:")
        val ruleList = listTopicRulesResponse.rules
        ruleList?.forEach { rule ->
            println("Rule name: ${rule.ruleName}")
            println("Rule ARN: ${rule.ruleArn}")
            println("--------------")
        }
    }
}

suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
```

```
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}

suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}

suspend fun listCertificates() {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}

suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}

private fun getValue(input: String?): String {
```

```
    // Define a regular expression pattern for extracting the subdomain.
    val pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\\.com")

    // Match the pattern against the input string.
    val matcher = pattern.matcher(input)

    // Check if a match is found.
    if (matcher.find()) {
        val subdomain = matcher.group(1)
        println("Extracted subdomain: $subdomain")
        return subdomain
    } else {
        println("No match found")
    }
    return ""
}

suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}

suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
```

```
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }

    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}

suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}

suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
```

```
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}

suspend fun createCertificate(): String? {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}

suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

작업

`AttachThingPrincipal`

다음 코드 예시에서는 AttachThingPrincipal을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 AttachThingPrincipal을 참조하세요.

`CreateKeysAndCertificate`

다음 코드 예시에서는 `CreateKeysAndCertificate`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createCertificate(): String? {
```

```
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateKeysAndCertificate를 참조하세요.

`CreateThing`

다음 코드 예시에서는 `CreateThing`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateThing을 참조하세요.

`CreateTopicRule`

다음 코드 예시에서는 `CreateTopicRule`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }
```

```
    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateTopicRule을 참조하세요.

`DeleteCertificate`

다음 코드 예시에서는 `DeleteCertificate`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteCertificate를 참조하세요.

`DeleteThing`

다음 코드 예시에서는 `DeleteThing`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteThing을 참조하세요.

`DescribeEndpoint`

다음 코드 예시에서는 `DescribeEndpoint`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
```

```
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeEndpoint를 참조하세요.

`DescribeThing`

다음 코드 예시에서는 `DescribeThing`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeThing을 참조하세요.

`DetachThingPrincipal`

다음 코드 예시에서는 `DetachThingPrincipal`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DetachThingPrincipal을 참조하세요.

`ListCertificates`

다음 코드 예시에서는 `ListCertificates`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listCertificates() {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListCertificates를 참조하세요.

`SearchIndex`

다음 코드 예시에서는 `SearchIndex`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }
```

```
    IotClient { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 SearchIndex을 참조하세요.

`UpdateThing`

다음 코드 예시에서는 `UpdateThing`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
```

```
            attributePayload = attributePayloadVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 UpdateThing을 참조하세요.

AWS IoT data SDK for Kotlin을 사용한 예제

다음 코드 예제에서는 Kotlin용 AWS SDK를와 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS IoT data.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`GetThingShadow`

다음 코드 예시에서는 `GetThingShadow`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetThingShadow를 참조하세요.

`UpdateThingShadow`

다음 코드 예시에서는 `UpdateThingShadow`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
```

```
        }

    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 UpdateThingShadow를 참조하세요.

SDK for Kotlin을 사용한 Amazon Keyspaces 예제

다음 코드 예제에서는 Amazon Keyspaces와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시작

Hello Amazon Keyspaces

다음 코드 예제에서는 Amazon Keyspaces를 사용하여 시작하는 방법을 보여줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
```

```
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
*/

suspend fun main() {
    listKeyspaces()
}

suspend fun listKeyspaces() {
    val keyspacesRequest =
        ListKeyspacesRequest {
            maxResults = 10
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.listKeyspaces(keyspacesRequest)
        response.keyspaces?.forEach { keyspace ->
            println("The name of the keyspace is ${keyspace.keyspaceName}")
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Kotlin API reference의 ListKeyspaces를 참조하세요.

주제

- 기본 사항
- 작업

기본 사항

기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 키스페이스와 테이블을 생성하세요. 테이블 스키마에는 영화 데이터가 저장되며 특정 시점으로 복구가 활성화되어 있습니다.

- SiGV4 인증을 통한 보안 TLS 연결을 사용하여 키스페이스에 연결합니다.
- 테이블을 쿼리합니다. 영화 데이터를 추가, 검색 및 업데이트합니다.
- 테이블을 업데이트 하세요. 열을 추가하여 시청한 영화를 추적합니다.
- 테이블을 이전 상태로 복원하고 리소스를 정리합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:

 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This example uses a secure file format to hold certificate information for
 Kotlin applications. This is required to make a connection to Amazon Keyspaces.
 For more information, see the following documentation topic:

 https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html

 This Kotlin example performs the following tasks:

 1. Create a keyspace.
 2. Check for keyspace existence.
 3. List keyspaces using a paginator.
 4. Create a table with a simple movie data schema and enable point-in-time
 recovery.
 5. Check for the table to be in an Active state.
 6. List all tables in the keyspace.
 7. Use a Cassandra driver to insert some records into the Movie table.
 8. Get all records from the Movie table.
 9. Get a specific Movie.
```

```
 10. Get a UTC timestamp for the current time.
 11. Update the table schema to add a ‘watched’ Boolean column.
 12. Update an item as watched.
 13. Query for items with watched = True.
 14. Restore the table back to the previous state using the timestamp.
 15. Check for completion of the restore action.
 16. Delete the table.
 17. Confirm that both tables are deleted.
 18. Delete the keyspace.
 */

/*
   Usage:
     fileName - The name of the JSON file that contains movie data. (Get this file
 from the GitHub repo at resources/sample_file.)
     keyspaceName - The name of the keyspace to create.
 */
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main() {
    val fileName = "<Replace with the JSON file that contains movie data>"
    val keyspaceName = "<Replace with the name of the keyspace to create>"
    val titleUpdate = "The Family"
    val yearUpdate = 2013
    val tableName = "MovieKotlin"
    val tableNameRestore = "MovieRestore"

    val loader = DriverConfigLoader.fromClasspath("application.conf")
    val session =
        CqlSession
            .builder()
            .withConfigLoader(loader)
            .build()

    println(DASHES)
    println("Welcome to the Amazon Keyspaces example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create a keyspace.")
    createKeySpace(keyspaceName)
    println(DASHES)

    println(DASHES)
```

```
    delay(5000)
    println("2. Check for keyspace existence.")
    checkKeyspaceExistence(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("3. List keyspaces using a paginator.")
    listKeyspacesPaginator()
    println(DASHES)

    println(DASHES)
    println("4. Create a table with a simple movie data schema and enable point-in-
time recovery.")
    createTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("5. Check for the table to be in an Active state.")
    delay(6000)
    checkTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("6. List all tables in the keyspace.")
    listTables(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("7. Use a Cassandra driver to insert some records into the Movie
 table.")
    delay(6000)
    loadData(session, fileName, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("8. Get all records from the Movie table.")
    getMovieData(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("9. Get a specific Movie.")
    getSpecificMovie(session, keyspaceName)
    println(DASHES)
```

```
    println(DASHES)
    println("10. Get a UTC timestamp for the current time.")
    val utc = ZonedDateTime.now(ZoneOffset.UTC)
    println("DATETIME = ${Date.from(utc.toInstant())}")
    println(DASHES)

    println(DASHES)
    println("11. Update the table schema to add a watched Boolean column.")
    updateTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("12. Update an item as watched.")
    delay(10000) // Wait 10 seconds for the update.
    updateRecord(session, keyspaceName, titleUpdate, yearUpdate)
    println(DASHES)

    println(DASHES)
    println("13. Query for items with watched = True.")
    getWatchedData(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("14. Restore the table back to the previous state using the timestamp.")
    println("Note that the restore operation can take up to 20 minutes.")
    restoreTable(keyspaceName, utc)
    println(DASHES)

    println(DASHES)
    println("15. Check for completion of the restore action.")
    delay(5000)
    checkRestoredTable(keyspaceName, "MovieRestore")
    println(DASHES)

    println(DASHES)
    println("16. Delete both tables.")
    deleteTable(keyspaceName, tableName)
    deleteTable(keyspaceName, tableNameRestore)
    println(DASHES)

    println(DASHES)
    println("17. Confirm that both tables are deleted.")
    checkTableDelete(keyspaceName, tableName)
    checkTableDelete(keyspaceName, tableNameRestore)
```

```
    println(DASHES)

    println(DASHES)
    println("18. Delete the keyspace.")
    deleteKeyspace(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("The scenario has completed successfully.")
    println(DASHES)
}

suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}

suspend fun checkTableDelete(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var status: String
    var response: GetTableResponse
    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    try {
        KeyspacesClient { region = "us-east-1" }.use { keyClient ->
            // Keep looping until the table cannot be found and a
 ResourceNotFoundException is thrown.
            while (true) {
                response = keyClient.getTable(tableRequest)
                status = response.status.toString()
                println(". The table status is $status")
                delay(500)
```

```
            }
        }
    } catch (e: ResourceNotFoundException) {
        println(e.message)
    }
    println("The table is deleted")
}

suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}

suspend fun checkRestoredTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println("The table status is $status")

            if (status.compareTo("ACTIVE") == 0) {
```

```
                tableStatus = true
            }
            delay(500)
        }

        val cols = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}

fun getWatchedData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin\"
 WHERE watched = true ALLOW FILTERING;")
```

```
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

fun updateRecord(
    session: CqlSession,
    keySpace: String,
    titleUpdate: String?,
    yearUpdate: Int,
) {
    val sqlStatement =
        "UPDATE \"$keySpace\".\"MovieKotlin\" SET watched=true WHERE title = :k0 AND
 year = :k1;"
    val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
    builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
    val preparedStatement = session.prepare(sqlStatement)
    builder.addStatement(
        preparedStatement
            .boundStatementBuilder()
            .setString("k0", titleUpdate)
            .setInt("k1", yearUpdate)
            .build(),
    )
    val batchStatement = builder.build()
    session.execute(batchStatement)
}

suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
```

```
            addColumns = listOf(def)
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}

fun getSpecificMovie(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet =
        session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin\" WHERE title
 = 'The Family' ALLOW FILTERING ;")

    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Get records from the Movie table.
fun getMovieData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin
\";")
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Load data into the table.
fun loadData(
    session: CqlSession,
    fileName: String,
    keySpace: String,
) {
    val sqlStatement =
```

```
        "INSERT INTO \"$keySpace\".\"MovieKotlin\" (title, year, plot) values
 (:k0, :k1, :k2)"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        // Insert the data into the Amazon Keyspaces table.
        val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
        val preparedStatement: PreparedStatement = session.prepare(sqlStatement)
        builder.addStatement(
            preparedStatement
                .boundStatementBuilder()
                .setString("k0", title)
                .setInt("k1", year)
                .setString("k2", info)
                .build(),
        )

        val batchStatement = builder.build()
        session.execute(batchStatement)
        t++
    }
}

suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
```

```
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}

suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

suspend fun createTable(
    keySpaceVal: String?,
```

```
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
    colList.add(defYear)
    colList.add(defReleaseDate)
    colList.add(defPlot)

    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }

    val titleKey =
        PartitionKey {
            name = "title"
        }
```

```
    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)

    val schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}

suspend fun listKeyspacesPaginator() {
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}

suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
```

```
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse = keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}

suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

작업

`CreateKeyspace`

다음 코드 예시에서는 `CreateKeyspace`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- API 세부 정보는 Kotlin용AWS SDK API 참조의 CreateKeyspace를 참조하세요.

`CreateTable`

다음 코드 예시에서는 `CreateTable`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
```

```
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
    colList.add(defYear)
    colList.add(defReleaseDate)
    colList.add(defPlot)

    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }

    val titleKey =
        PartitionKey {
            name = "title"
        }

    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)
```

```
    val schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateTable를 참조하세요.

`DeleteKeyspace`

다음 코드 예시에서는 `DeleteKeyspace`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteKeyspace(keyspaceNameVal: String?) {
```

```
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}
```

- API 세부 정보는 Kotlin용AWS SDK API 참조의 DeleteKeyspace를 참조하세요.

`DeleteTable`

다음 코드 예시에서는 `DeleteTable`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteTable를 참조하세요.

`GetKeyspace`

다음 코드 예시에서는 `GetKeyspace`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse = keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}
```

- API에 대한 세부 정보는 Kotlin용AWS SDK API 참조의 GetKeyspace를 참조하세요.

`GetTable`

다음 코드 예시에서는 `GetTable`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}
```

- API 세부 정보는 Kotlin용AWS SDK API 참조의 GetTable을 참조하세요.

`ListKeyspaces`

다음 코드 예시에서는 `ListKeyspaces`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listKeyspacesPaginator() {
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Kotlin API reference의 ListKeyspaces를 참조하세요.

`ListTables`

다음 코드 예시에서는 `ListTables`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }
```

```
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListTables를 참조하세요.

`RestoreTable`

다음 코드 예시에서는 `RestoreTable`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
```

```
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API reference의 RestoreTable을 참조하세요.

`UpdateTable`

다음 코드 예시에서는 `UpdateTable`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
```

```
        keyClient.updateTable(tableRequest)
    }
}
```

- API에 대한 세부 정보는 AWS Kotlin용 SDK API 참조의 UpdateTable을 참조하세요.

AWS KMS SDK for Kotlin을 사용한 예제

다음 코드 예제에서는 Kotlin용 AWS SDK를와 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS KMS.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`CreateAlias`

다음 코드 예시에서는 `CreateAlias`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createCustomAlias(
    targetKeyIdVal: String?,
    aliasNameVal: String?,
```

```
) {
    val request =
        CreateAliasRequest {
            aliasName = aliasNameVal
            targetKeyId = targetKeyIdVal
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        kmsClient.createAlias(request)
        println("$aliasNameVal was successfully created")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateAlias를 참조하십시오.

`CreateGrant`

다음 코드 예시에서는 `CreateGrant`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createNewGrant(
    keyIdVal: String?,
    granteePrincipalVal: String?,
    operation: String,
): String? {
    val operationOb = GrantOperation.fromValue(operation)
    val grantOperationList = ArrayList<GrantOperation>()
    grantOperationList.add(operationOb)

    val request =
        CreateGrantRequest {
            keyId = keyIdVal
            granteePrincipal = granteePrincipalVal
```

```
            operations = grantOperationList
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.createGrant(request)
        return response.grantId
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateGrant를 참조하십시오.

`CreateKey`

다음 코드 예시에서는 `CreateKey`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createKey(keyDesc: String?): String? {
    val request =
        CreateKeyRequest {
            description = keyDesc
            customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
            keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val result = kmsClient.createKey(request)
        println("Created a customer key with id " + result.keyMetadata?.arn)
        return result.keyMetadata?.keyId
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateKey를 참조하십시오.

`Decrypt`

다음 코드 예시에서는 `Decrypt`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
```

```
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
        print(myVal)
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Decrypt를 참조하십시오.

`DescribeKey`

다음 코드 예시에서는 `DescribeKey`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeSpecifcKey(keyIdVal: String?) {
    val request =
        DescribeKeyRequest {
            keyId = keyIdVal
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.describeKey(request)
        println("The key description is ${response.keyMetadata?.description}")
        println("The key ARN is ${response.keyMetadata?.arn}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeKey를 참조하십시오.

`DisableKey`

다음 코드 예시에서는 `DisableKey`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun disableKey(keyIdVal: String?) {
    val request =
        DisableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        kmsClient.disableKey(request)
        println("$keyIdVal was successfully disabled")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DisableKey을 참조하십시오.

`EnableKey`

다음 코드 예시에서는 `EnableKey`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun enableKey(keyIdVal: String?) {
    val request =
        EnableKeyRequest {
            keyId = keyIdVal
        }
```

```
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        kmsClient.enableKey(request)
        println("$keyIdVal was successfully enabled.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 EnableKey를 참조하십시오.

`Encrypt`

다음 코드 예시에서는 `Encrypt`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}
```

```
suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
        print(myVal)
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Encrypt를 참조하십시오.

`ListAliases`

다음 코드 예시에서는 `ListAliases`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listAllAliases() {
    val request =
        ListAliasesRequest {
            limit = 15
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listAliases(request)
        response.aliases?.forEach { alias ->
```

```
            println("The alias name is ${alias.aliasName}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListAliases를 참조하십시오.

`ListGrants`

다음 코드 예시에서는 `ListGrants`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun displayGrantIds(keyIdVal: String?) {
    val request =
        ListGrantsRequest {
            keyId = keyIdVal
            limit = 15
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listGrants(request)
        response.grants?.forEach { grant ->
            println("The grant Id is ${grant.grantId}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListGrants를 참조하십시오.

`ListKeys`

다음 코드 예시에서는 `ListKeys`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listAllKeys() {
    val request =
        ListKeysRequest {
            limit = 15
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listKeys(request)
        response.keys?.forEach { key ->
            println("The key ARN is ${key.keyArn}")
            println("The key Id is ${key.keyId}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListKeys를 참조하십시오.

SDK for Kotlin을 사용한 Lambda 예제

다음 코드 예제에서는 Lambda와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 전체 소스 코드에 대한 링크가 포함되어 있습니다.

주제

- 기본 사항
- 작업
- 시나리오

기본 사항

기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- IAM 역할과 Lambda 함수를 생성하고 핸들러 코드를 업로드합니다.
- 단일 파라미터로 함수를 간접적으로 간접 호출하고 결과를 가져옵니다.
- 함수 코드를 업데이트하고 환경 변수로 구성합니다.
- 새 파라미터로 함수를 간접적으로 간접 호출하고 결과를 가져옵니다. 반환된 실행 로그를 표시합니다.
- 계정의 함수를 나열합니다.

자세한 내용은 콘솔로 Lambda 함수 생성을 참조하십시오.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <functionName> <role> <handler> <bucketName> <updatedBucketName> <key>

        Where:
            functionName - The name of the AWS Lambda function.
            role - The AWS Identity and Access Management (IAM) service role that
 has AWS Lambda permissions.
```

```
            handler - The fully qualified method name (for example,
example.Handler::handleRequest).
            bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name
that contains the ZIP or JAR used for the Lambda function's code.
            updatedBucketName - The Amazon S3 bucket name that contains the .zip
or .jar used to update the Lambda function's code.
            key - The Amazon S3 key name that represents the .zip or .jar file (for
example, LambdaHello-1.0-SNAPSHOT.jar).
            """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val functionName = args[0]
    val role = args[1]
    val handler = args[2]
    val bucketName = args[3]
    val updatedBucketName = args[4]
    val key = args[5]

    println("Creating a Lambda function named $functionName.")
    val funArn = createScFunction(functionName, bucketName, key, handler, role)
    println("The AWS Lambda ARN is $funArn")

    // Get a specific Lambda function.
    println("Getting the $functionName AWS Lambda function.")
    getFunction(functionName)

    // List the Lambda functions.
    println("Listing all AWS Lambda functions.")
    listFunctionsSc()

    // Invoke the Lambda function.
    println("*** Invoke the Lambda function.")
    invokeFunctionSc(functionName)

    // Update the AWS Lambda function code.
    println("*** Update the Lambda function code.")
    updateFunctionCode(functionName, updatedBucketName, key)

    // println("*** Invoke the function again after updating the code.")
    invokeFunctionSc(functionName)
```

```
    // Update the AWS Lambda function configuration.
    println("Update the run time of the function.")
    updateFunctionConfiguration(functionName, handler)

    // Delete the AWS Lambda function.
    println("Delete the AWS Lambda function.")
    delFunction(functionName)
}

suspend fun createScFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String,
): String {
    val functionCode =
        FunctionCode {
            s3Bucket = s3BucketName
            s3Key = myS3Key
        }

    val request =
        CreateFunctionRequest {
            functionName = myFunctionName
            code = functionCode
            description = "Created by the Lambda Kotlin API"
            handler = myHandler
            role = myRole
            runtime = Runtime.Java17
        }

    // Create a Lambda function using a waiter
    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn.toString()
    }
}

suspend fun getFunction(functionNameVal: String) {
```

```
    val functionRequest =
        GetFunctionRequest {
            functionName = functionNameVal
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.getFunction(functionRequest)
        println("The runtime of this Lambda function is
 ${response.configuration?.runtime}")
    }
}

suspend fun listFunctionsSc() {
    val request =
        ListFunctionsRequest {
            maxItems = 10
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.listFunctions(request)
        response.functions?.forEach { function ->
            println("The function name is ${function.functionName}")
        }
    }
}

suspend fun invokeFunctionSc(functionNameVal: String) {
    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request =
        InvokeRequest {
            functionName = functionNameVal
            payload = byteArray
            logType = LogType.Tail
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("The function payload is ${res.payload?.toString(Charsets.UTF_8)}")
    }
}

suspend fun updateFunctionCode(
    functionNameVal: String?,
```

```
    bucketName: String?,
    key: String?,
) {
    val functionCodeRequest =
        UpdateFunctionCodeRequest {
            functionName = functionNameVal
            publish = true
            s3Bucket = bucketName
            s3Key = key
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.updateFunctionCode(functionCodeRequest)
        awsLambda.waitUntilFunctionUpdated {
            functionName = functionNameVal
        }
        println("The last modified value is " + response.lastModified)
    }
}

suspend fun updateFunctionConfiguration(
    functionNameVal: String?,
    handlerVal: String?,
) {
    val configurationRequest =
        UpdateFunctionConfigurationRequest {
            functionName = functionNameVal
            handler = handlerVal
            runtime = Runtime.Java17
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.updateFunctionConfiguration(configurationRequest)
    }
}

suspend fun delFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
```

```
        println("$myFunctionName was deleted")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - 간접 호출
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

작업

`CreateFunction`

다음 코드 예시에서는 `CreateFunction`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createNewFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String,
): String? {
    val functionCode =
        FunctionCode {
            s3Bucket = s3BucketName
```

```
            s3Key = myS3Key
        }

    val request =
        CreateFunctionRequest {
            functionName = myFunctionName
            code = functionCode
            description = "Created by the Lambda Kotlin API"
            handler = myHandler
            role = myRole
            runtime = Runtime.Java17
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateFunction를 참조하십시오.

`DeleteFunction`

다음 코드 예시에서는 `DeleteFunction`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun delLambdaFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }
```

```
    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteFunction를 참조하십시오.

`Invoke`

다음 코드 예시에서는 `Invoke`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun invokeFunction(functionNameVal: String) {
    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request =
        InvokeRequest {
            functionName = functionNameVal
            logType = LogType.Tail
            payload = byteArray
        }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("${res.payload?.toString(Charsets.UTF_8)}")
        println("The log result is ${res.logResult}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Invoke를 참조하십시오.

시나리오

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예시에서 사용되는 서비스

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

SDK for Kotlin을 사용한 MediaConvert 예제

다음 코드 예제에서는 AWS MediaConvert와 함께 SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`CreateJob`

다음 코드 예시에서는 `CreateJob`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createMediaJob(
    mcClient: MediaConvertClient,
    mcRoleARN: String,
    fileInputVal: String,
): String? {
    val s3path = fileInputVal.substring(0, fileInputVal.lastIndexOf('/') + 1) +
 "javasdk/out/"
    val fileOutput = s3path + "index"
    val thumbsOutput = s3path + "thumbs/"
    val mp4Output = s3path + "mp4/"

    try {
        val describeEndpoints =
            DescribeEndpointsRequest {
                maxResults = 20
            }

        val res = mcClient.describeEndpoints(describeEndpoints)
        if (res.endpoints?.size!! <= 0) {
            println("Cannot find MediaConvert service endpoint URL!")
            exitProcess(0)
        }
        val endpointURL = res.endpoints!!.get(0).url!!
        val mediaConvert =
            MediaConvertClient.fromEnvironment {
                region = "us-west-2"
                endpointProvider =
                    MediaConvertEndpointProvider {
```

```
                        Endpoint(endpointURL)
                    }
            }

        // output group Preset HLS low profile
        val hlsLow = createOutput("_low", "_\$dt$", 750000, 7, 1920, 1080, 640)

        // output group Preset HLS medium profile
        val hlsMedium = createOutput("_medium", "_\$dt$", 1200000, 7, 1920, 1080,
1280)

        // output group Preset HLS high profole
        val hlsHigh = createOutput("_high", "_\$dt$", 3500000, 8, 1920, 1080, 1920)

        val outputSettings =
            OutputGroupSettings {
                type = OutputGroupType.HlsGroupSettings
            }

        val outputObsList: MutableList<Output> = mutableListOf()
        if (hlsLow != null) {
            outputObsList.add(hlsLow)
        }
        if (hlsMedium != null) {
            outputObsList.add(hlsMedium)
        }
        if (hlsHigh != null) {
            outputObsList.add(hlsHigh)
        }

        // Create an OutputGroup object.
        val appleHLS =
            OutputGroup {
                name = "Apple HLS"
                customName = "Example"
                outputGroupSettings =
                    OutputGroupSettings {
                        type = OutputGroupType.HlsGroupSettings
                        this.hlsGroupSettings =
                            HlsGroupSettings {
                                directoryStructure =
HlsDirectoryStructure.SingleDirectory
                                manifestDurationFormat =
HlsManifestDurationFormat.Integer
```

```
                                streamInfResolution = HlsStreamInfResolution.Include
                                clientCache = HlsClientCache.Enabled
                                captionLanguageSetting =
HlsCaptionLanguageSetting.Omit
                                manifestCompression = HlsManifestCompression.None
                                codecSpecification = HlsCodecSpecification.Rfc4281
                                outputSelection =
HlsOutputSelection.ManifestsAndSegments
                                programDateTime = HlsProgramDateTime.Exclude
                                programDateTimePeriod = 600
                                timedMetadataId3Frame =
HlsTimedMetadataId3Frame.Priv
                                timedMetadataId3Period = 10
                                destination = fileOutput
                                segmentControl = HlsSegmentControl.SegmentedFiles
                                minFinalSegmentLength = 0.toDouble()
                                segmentLength = 4
                                minSegmentLength = 1
                            }
                    }
                outputs = outputObsList
            }

        val theOutput =
            Output {
                extension = "mp4"
                containerSettings =
                    ContainerSettings {
                        container = ContainerType.fromValue("MP4")
                    }

                videoDescription =
                    VideoDescription {
                        width = 1280
                        height = 720
                        scalingBehavior = ScalingBehavior.Default
                        sharpness = 50
                        antiAlias = AntiAlias.Enabled
                        timecodeInsertion = VideoTimecodeInsertion.Disabled
                        colorMetadata = ColorMetadata.Insert
                        respondToAfd = RespondToAfd.None
                        afdSignaling = AfdSignaling.None
                        dropFrameTimecode = DropFrameTimecode.Enabled
                        codecSettings =
```

```
                            VideoCodecSettings {
                                codec = VideoCodec.H264
                                h264Settings =
                                    H264Settings {
                                        rateControlMode = H264RateControlMode.Qvbr
                                        parControl =
H264ParControl.InitializeFromSource
                                        qualityTuningLevel =
H264QualityTuningLevel.SinglePass
                                        qvbrSettings = H264QvbrSettings
{ qvbrQualityLevel = 8 }
                                        codecLevel = H264CodecLevel.Auto
                                        codecProfile = H264CodecProfile.Main
                                        maxBitrate = 2400000
                                        framerateControl =
H264FramerateControl.InitializeFromSource
                                        gopSize = 2.0
                                        gopSizeUnits = H264GopSizeUnits.Seconds
                                        numberBFramesBetweenReferenceFrames = 2
                                        gopClosedCadence = 1
                                        gopBReference = H264GopBReference.Disabled
                                        slowPal = H264SlowPal.Disabled
                                        syntax = H264Syntax.Default
                                        numberReferenceFrames = 3
                                        dynamicSubGop = H264DynamicSubGop.Static
                                        fieldEncoding = H264FieldEncoding.Paff
                                        sceneChangeDetect =
H264SceneChangeDetect.Enabled
                                        minIInterval = 0
                                        telecine = H264Telecine.None
                                        framerateConversionAlgorithm =
H264FramerateConversionAlgorithm.DuplicateDrop
                                        entropyEncoding = H264EntropyEncoding.Cabac
                                        slices = 1
                                        unregisteredSeiTimecode =
H264UnregisteredSeiTimecode.Disabled
                                        repeatPps = H264RepeatPps.Disabled
                                        adaptiveQuantization =
H264AdaptiveQuantization.High
                                        spatialAdaptiveQuantization =
H264SpatialAdaptiveQuantization.Enabled
                                        temporalAdaptiveQuantization =
H264TemporalAdaptiveQuantization.Enabled
```

```
                                        flickerAdaptiveQuantization =
H264FlickerAdaptiveQuantization.Disabled
                                        softness = 0
                                        interlaceMode =
H264InterlaceMode.Progressive
                                    }
                                }
                            }

                        audioDescriptions =
                            listOf(
                                AudioDescription {
                                    audioTypeControl = AudioTypeControl.FollowInput
                                    languageCodeControl =
AudioLanguageCodeControl.FollowInput
                                    codecSettings =
                                        AudioCodecSettings {
                                            codec = AudioCodec.Aac
                                            aacSettings =
                                                AacSettings {
                                                    codecProfile = AacCodecProfile.Lc
                                                    rateControlMode = AacRateControlMode.Cbr
                                                    codingMode = AacCodingMode.CodingMode2_0
                                                    sampleRate = 44100
                                                    bitrate = 160000
                                                    rawFormat = AacRawFormat.None
                                                    specification = AacSpecification.Mpeg4
                                                    audioDescriptionBroadcasterMix =
AacAudioDescriptionBroadcasterMix.Normal
                                                }
                                        }
                                },
                            )
                    }

                // Create an OutputGroup
                val fileMp4 =
                    OutputGroup {
                        name = "File Group"
                        customName = "mp4"
                        outputGroupSettings =
                            OutputGroupSettings {
                                type = OutputGroupType.FileGroupSettings
                                fileGroupSettings =
```

```
                            FileGroupSettings {
                                destination = mp4Output
                            }
                    }
                outputs = listOf(theOutput)
            }

        val containerSettings1 =
            ContainerSettings {
                container = ContainerType.Raw
            }

        val thumbs =
            OutputGroup {
                name = "File Group"
                customName = "thumbs"
                outputGroupSettings =
                    OutputGroupSettings {
                        type = OutputGroupType.FileGroupSettings
                        fileGroupSettings =
                            FileGroupSettings {
                                destination = thumbsOutput
                            }
                    }

                outputs =
                    listOf(
                        Output {
                            extension = "jpg"

                            this.containerSettings = containerSettings1
                            videoDescription =
                                VideoDescription {
                                    scalingBehavior = ScalingBehavior.Default
                                    sharpness = 50
                                    antiAlias = AntiAlias.Enabled
                                    timecodeInsertion =
VideoTimecodeInsertion.Disabled
                                    colorMetadata = ColorMetadata.Insert
                                    dropFrameTimecode = DropFrameTimecode.Enabled
                                    codecSettings =
                                        VideoCodecSettings {
                                            codec = VideoCodec.FrameCapture
                                            frameCaptureSettings =
```

```
                                        FrameCaptureSettings {
                                            framerateNumerator = 1
                                            framerateDenominator = 1
                                            maxCaptures = 10000000
                                            quality = 80
                                        }
                                    }
                                }
                            },
                        )
                }

            val audioSelectors1: MutableMap<String, AudioSelector> = HashMap()
            audioSelectors1["Audio Selector 1"] =
                AudioSelector {
                    defaultSelection = AudioDefaultSelection.Default
                    offset = 0
                }

            val jobSettings =
                JobSettings {
                    inputs =
                        listOf(
                            Input {
                                audioSelectors = audioSelectors1
                                videoSelector =
                                    VideoSelector {
                                        colorSpace = ColorSpace.Follow
                                        rotate = InputRotate.Degree0
                                    }
                                filterEnable = InputFilterEnable.Auto
                                filterStrength = 0
                                deblockFilter = InputDeblockFilter.Disabled
                                denoiseFilter = InputDenoiseFilter.Disabled
                                psiControl = InputPsiControl.UsePsi
                                timecodeSource = InputTimecodeSource.Embedded
                                fileInput = fileInputVal

                                outputGroups = listOf(appleHLS, thumbs, fileMp4)
                            },
                        )
                }

            val createJobRequest =
```

```
            CreateJobRequest {
                role = mcRoleARN
                settings = jobSettings
            }

        val createJobResponse = mediaConvert.createJob(createJobRequest)
        return createJobResponse.job?.id
    } catch (ex: MediaConvertException) {
        println(ex.message)
        mcClient.close()
        exitProcess(0)
    }
}

fun createOutput(
    nameModifierVal: String,
    segmentModifierVal: String,
    qvbrMaxBitrate: Int,
    qvbrQualityLevelVal: Int,
    originWidth: Int,
    originHeight: Int,
    targetWidth: Int,
): Output? {
    val targetHeight = (
        (originHeight * targetWidth / originWidth).toFloat().roundToInt() -
            (originHeight * targetWidth / originWidth).toFloat().roundToInt() % 4
        )

    var output: Output?
    try {
        val audio1 =
            AudioDescription {
                audioTypeControl = AudioTypeControl.FollowInput
                languageCodeControl = AudioLanguageCodeControl.FollowInput
                codecSettings =
                    AudioCodecSettings {
                        codec = AudioCodec.Aac
                        aacSettings =
                            AacSettings {
                                codecProfile = AacCodecProfile.Lc
                                rateControlMode = AacRateControlMode.Cbr
                                codingMode = AacCodingMode.CodingMode2_0
                                sampleRate = 44100
                                bitrate = 96000
```

```
                                rawFormat = AacRawFormat.None
                                specification = AacSpecification.Mpeg4
                                audioDescriptionBroadcasterMix =
AacAudioDescriptionBroadcasterMix.Normal
                            }
                        }
                    }

                output =
                    Output {
                        nameModifier = nameModifierVal
                        outputSettings =
                            OutputSettings {
                                hlsSettings =
                                    HlsSettings {
                                        segmentModifier = segmentModifierVal
                                        audioGroupId = "program_audio"
                                        iFrameOnlyManifest = HlsIFrameOnlyManifest.Exclude
                                    }
                            }
                        containerSettings =
                            ContainerSettings {
                                container = ContainerType.M3U8
                                this.m3u8Settings =
                                    M3u8Settings {
                                        audioFramesPerPes = 4
                                        pcrControl = M3u8PcrControl.PcrEveryPesPacket
                                        pmtPid = 480
                                        privateMetadataPid = 503
                                        programNumber = 1
                                        patInterval = 0
                                        pmtInterval = 0
                                        scte35Source = M3u8Scte35Source.None
                                        scte35Pid = 500
                                        nielsenId3 = M3u8NielsenId3.None
                                        timedMetadata = TimedMetadata.None
                                        timedMetadataPid = 502
                                        videoPid = 481
                                        audioPids = listOf(482, 483, 484, 485, 486, 487,
488, 489, 490, 491, 492)
                                    }

                                videoDescription =
                                    VideoDescription {
```

```
                                width = targetWidth
                                height = targetHeight
                                scalingBehavior = ScalingBehavior.Default
                                sharpness = 50
                                antiAlias = AntiAlias.Enabled
                                timecodeInsertion = VideoTimecodeInsertion.Disabled
                                colorMetadata = ColorMetadata.Insert
                                respondToAfd = RespondToAfd.None
                                afdSignaling = AfdSignaling.None
                                dropFrameTimecode = DropFrameTimecode.Enabled
                                codecSettings =
                                    VideoCodecSettings {
                                        codec = VideoCodec.H264
                                        h264Settings =
                                            H264Settings {
                                                rateControlMode =
H264RateControlMode.Qvbr
                                                parControl =
H264ParControl.InitializeFromSource
                                                qualityTuningLevel =
H264QualityTuningLevel.SinglePass
                                                qvbrSettings =
                                                    H264QvbrSettings {
                                                        qvbrQualityLevel =
qvbrQualityLevelVal
                                                    }
                                                codecLevel = H264CodecLevel.Auto
                                                codecProfile =
                                                    if (targetHeight > 720 &&
                                                        targetWidth > 1280
                                                    ) {
                                                        H264CodecProfile.High
                                                    } else {
                                                        H264CodecProfile.Main
                                                    }
                                                maxBitrate = qvbrMaxBitrate
                                                framerateControl =
H264FramerateControl.InitializeFromSource
                                                gopSize = 2.0
                                                gopSizeUnits =
H264GopSizeUnits.Seconds
                                                numberBFramesBetweenReferenceFrames
= 2
                                                gopClosedCadence = 1
```

```
                                                gopBReference =
 H264GopBReference.Disabled
                                                slowPal = H264SlowPal.Disabled
                                                syntax = H264Syntax.Default
                                                numberReferenceFrames = 3
                                                dynamicSubGop =
 H264DynamicSubGop.Static
                                                fieldEncoding =
 H264FieldEncoding.Paff
                                                sceneChangeDetect =
 H264SceneChangeDetect.Enabled
                                                minIInterval = 0
                                                telecine = H264Telecine.None
                                                framerateConversionAlgorithm =
 H264FramerateConversionAlgorithm.DuplicateDrop
                                                entropyEncoding =
 H264EntropyEncoding.Cabac
                                                slices = 1
                                                unregisteredSeiTimecode =
 H264UnregisteredSeiTimecode.Disabled
                                                repeatPps = H264RepeatPps.Disabled
                                                adaptiveQuantization =
 H264AdaptiveQuantization.High
                                                spatialAdaptiveQuantization =
 H264SpatialAdaptiveQuantization.Enabled
                                                temporalAdaptiveQuantization =
 H264TemporalAdaptiveQuantization.Enabled
                                                flickerAdaptiveQuantization =
 H264FlickerAdaptiveQuantization.Disabled
                                                softness = 0
                                                interlaceMode =
 H264InterlaceMode.Progressive
                                            }
                                        }
                                    audioDescriptions = listOf(audio1)
                                }
                            }
                        }
    } catch (ex: MediaConvertException) {
        println(ex.toString())
        exitProcess(0)
    }
    return output
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateJob을 참조하십시오.

`GetJob`

다음 코드 예시에서는 `GetJob`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getSpecificJob(
    mcClient: MediaConvertClient,
    jobId: String?,
) {
    val describeEndpoints =
        DescribeEndpointsRequest {
            maxResults = 20
        }

    val res = mcClient.describeEndpoints(describeEndpoints)
    if (res.endpoints?.size!! <= 0) {
        println("Cannot find MediaConvert service endpoint URL!")
        exitProcess(0)
    }

    val endpointURL = res.endpoints!!.get(0).url!!
    val mediaConvert =
        MediaConvertClient.fromEnvironment {
            region = "us-west-2"
            endpointProvider =
                MediaConvertEndpointProvider {
                    Endpoint(endpointURL)
                }
        }
```

```
    val jobRequest =
        GetJobRequest {
            id = jobId
        }

    val response: GetJobResponse = mediaConvert.getJob(jobRequest)
    println("The ARN of the job is ${response.job?.arn}.")
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetJob을 참조하십시오.

`ListJobs`

다음 코드 예시에서는 `ListJobs`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listCompleteJobs(mcClient: MediaConvertClient) {
    val describeEndpoints =
        DescribeEndpointsRequest {
            maxResults = 20
        }

    val res = mcClient.describeEndpoints(describeEndpoints)
    if (res.endpoints?.size!! <= 0) {
        println("Cannot find MediaConvert service endpoint URL!")
        exitProcess(0)
    }
    val endpointURL = res.endpoints!![0].url!!
    val mediaConvert =
        MediaConvertClient.fromEnvironment {
            region = "us-west-2"
            endpointProvider =
                MediaConvertEndpointProvider {
                    Endpoint(endpointURL)
```

```
                }
        }

    val jobsRequest =
        ListJobsRequest {
            maxResults = 10
            status = JobStatus.fromValue("COMPLETE")
        }

    val jobsResponse = mediaConvert.listJobs(jobsRequest)
    val jobs = jobsResponse.jobs
    if (jobs != null) {
        for (job in jobs) {
            println("The JOB ARN is ${job.arn}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListJobs를 참조하십시오.

SDK for Kotlin을 사용한 Amazon Pinpoint 예제

다음 코드 예제에서는 Amazon Pinpoint와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`CreateApp`

다음 코드 예시에서는 `CreateApp`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createApplication(applicationName: String?): String? {
    val createApplicationRequestOb =
        CreateApplicationRequest {
            name = applicationName
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.createApp(
                CreateAppRequest {
                    createApplicationRequest = createApplicationRequestOb
                },
            )
        return result.applicationResponse?.id
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateApp을 참조하십시오.

`CreateCampaign`

다음 코드 예시에서는 `CreateCampaign`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createPinCampaign(
    appId: String,
    segmentIdVal: String,
) {
    val scheduleOb =
        Schedule {
            startTime = "IMMEDIATE"
        }

    val defaultMessageOb =
        Message {
            action = Action.OpenApp
            body = "My message body"
            title = "My message title"
        }

    val messageConfigurationOb =
        MessageConfiguration {
            defaultMessage = defaultMessageOb
        }

    val writeCampaign =
        WriteCampaignRequest {
            description = "My description"
            schedule = scheduleOb
            name = "MyCampaign"
            segmentId = segmentIdVal
            messageConfiguration = messageConfigurationOb
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result: CreateCampaignResponse =
            pinpoint.createCampaign(
                CreateCampaignRequest {
                    applicationId = appId
                    writeCampaignRequest = writeCampaign
                },
            )
        println("Campaign ID is ${result.campaignResponse?.id}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateCampaign를 참조하십시오.

`CreateSegment`

다음 코드 예시에서는 `CreateSegment`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createPinpointSegment(applicationIdVal: String?): String? {
    val segmentAttributes = mutableMapOf<String, AttributeDimension>()
    val myList = mutableListOf<String>()
    myList.add("Lakers")

    val atts =
        AttributeDimension {
            attributeType = AttributeType.Inclusive
            values = myList
        }

    segmentAttributes["Team"] = atts
    val recencyDimension =
        RecencyDimension {
            duration = Duration.fromValue("DAY_30")
            recencyType = RecencyType.fromValue("ACTIVE")
        }

    val segmentBehaviors =
        SegmentBehaviors {
            recency = recencyDimension
        }

    val segmentLocation = SegmentLocation {}
    val dimensionsOb =
        SegmentDimensions {
            attributes = segmentAttributes
            behavior = segmentBehaviors
```

```
            demographic = SegmentDemographics {}
            location = segmentLocation
        }

    val writeSegmentRequestOb =
        WriteSegmentRequest {
            name = "MySegment101"
            dimensions = dimensionsOb
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val createSegmentResult: CreateSegmentResponse =
            pinpoint.createSegment(
                CreateSegmentRequest {
                    applicationId = applicationIdVal
                    writeSegmentRequest = writeSegmentRequestOb
                },
            )
        println("Segment ID is ${createSegmentResult.segmentResponse?.id}")
        return createSegmentResult.segmentResponse?.id
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateSegment를 참조하십시오.

`DeleteApp`

다음 코드 예시에서는 `DeleteApp`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deletePinApp(appId: String?) {
    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result =
```

```
            pinpoint.deleteApp(
                DeleteAppRequest {
                    applicationId = appId
                },
            )
        val appName = result.applicationResponse?.name
        println("Application $appName has been deleted.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteApp를 참조하십시오.

`DeleteEndpoint`

다음 코드 예시에서는 `DeleteEndpoint`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deletePinEncpoint(
    appIdVal: String?,
    endpointIdVal: String?,
) {
    val deleteEndpointRequest =
        DeleteEndpointRequest {
            applicationId = appIdVal
            endpointId = endpointIdVal
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.deleteEndpoint(deleteEndpointRequest)
        val id = result.endpointResponse?.id
        println("The deleted endpoint is  $id")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteEndpoint를 참조하십시오.

`GetEndpoint`

다음 코드 예시에서는 `GetEndpoint`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun lookupPinpointEndpoint(
    appId: String?,
    endpoint: String?,
) {
    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.getEndpoint(
                GetEndpointRequest {
                    applicationId = appId
                    endpointId = endpoint
                },
            )
        val endResponse = result.endpointResponse

        // Uses the Google Gson library to pretty print the endpoint JSON.
        val gson: com.google.gson.Gson =
            GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting()
                .create()

        val endpointJson: String = gson.toJson(endResponse)
        println(endpointJson)
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetEndpoint를 참조하십시오.

`GetSegments`

다음 코드 예시에서는 `GetSegments`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listSegs(appId: String?) {
    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val response =
            pinpoint.getSegments(
                GetSegmentsRequest {
                    applicationId = appId
                },
            )
        response.segmentsResponse?.item?.forEach { segment ->
            println("Segement id is ${segment.id}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetSegments를 참조하십시오.

`SendMessages`

다음 코드 예시에서는 `SendMessages`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

val body: String =
    """
    Amazon Pinpoint test (AWS SDK for Kotlin)

    This email was sent through the Amazon Pinpoint Email API using the AWS SDK for
 Kotlin.

    """.trimIndent()

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <subject> <appId> <senderAddress> <toAddress>

    Where:
        subject - The email subject to use.
        senderAddress - The from address. This address has to be verified in Amazon
 Pinpoint in the region you're using to send email
        toAddress - The to address. This address has to be verified in Amazon
 Pinpoint in the region you're using to send email
    """

    if (args.size != 3) {
        println(usage)
        exitProcess(0)
```

```
    }

    val subject = args[0]
    val senderAddress = args[1]
    val toAddress = args[2]
    sendEmail(subject, senderAddress, toAddress)
}

suspend fun sendEmail(
    subjectVal: String?,
    senderAddress: String,
    toAddressVal: String,
) {
    var content =
        Content {
            data = body
        }

    val messageBody =
        Body {
            text = content
        }

    val subContent =
        Content {
            data = subjectVal
        }

    val message =
        Message {
            body = messageBody
            subject = subContent
        }

    val destinationOb =
        Destination {
            toAddresses = listOf(toAddressVal)
        }

    val emailContent =
        EmailContent {
            simple = message
        }
```

```
    val sendEmailRequest =
        SendEmailRequest {
            fromEmailAddress = senderAddress
            destination = destinationOb
            this.content = emailContent
        }

    PinpointEmailClient { region = "us-east-1" }.use { pinpointemail ->
        pinpointemail.sendEmail(sendEmailRequest)
        println("Message Sent")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 SendMessages를 참조하십시오.

SDK for Kotlin을 사용한 Amazon RDS 예제

다음 코드 예제에서는 Amazon RDS와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 전체 소스 코드에 대한 링크가 포함되어 있습니다.

주제

- 기본 사항
- 작업
- 시나리오

기본 사항

기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용자 지정 DB 파라미터 그룹을 생성하고 파라미터 값을 설정합니다.
- 파라미터 그룹을 사용하도록 구성된 DB 인스턴스를 생성합니다. DB 인스턴스에는 데이터베이스도 포함되어 있습니다.
- 인스턴스의 스냅샷을 만듭니다.
- 인스턴스 및 파라미터 그룹을 삭제합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
Before running this code example, set up your development environment, including
 your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html

This example performs the following tasks:

1. Returns a list of the available DB engines by invoking the
 DescribeDbEngineVersions method.
```

```
2. Selects an engine family and create a custom DB parameter group by invoking the
 createDBParameterGroup method.
3. Gets the parameter groups by invoking the DescribeDbParameterGroups method.
4. Gets parameters in the group by invoking the DescribeDbParameters method.
5. Modifies both the auto_increment_offset and auto_increment_increment parameters
 by invoking the modifyDbParameterGroup method.
6. Gets and displays the updated parameters.
7. Gets a list of allowed engine versions by invoking the describeDbEngineVersions
 method.
8. Gets a list of micro instance classes available for the selected engine.
9. Creates an Amazon Relational Database Service (Amazon RDS) database instance that
 contains a MySQL database and uses the parameter group.
10. Waits for DB instance to be ready and prints out the connection endpoint value.
11. Creates a snapshot of the DB instance.
12. Waits for the DB snapshot to be ready.
13. Deletes the DB instance.
14. Deletes the parameter group.
 */

var sleepTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier> <dbName>
 <dbSnapshotIdentifier><secretName>

        Where:
            dbGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
 the database credentials.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val dbGroupName = args[0]
    val dbParameterGroupFamily = args[1]
```

```
    val dbInstanceIdentifier = args[2]
    val dbName = args[3]
    val dbSnapshotIdentifier = args[4]
    val secretName = args[5]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeDBEngines()

    println("2. Create a custom parameter group")
    createDBParameterGroup(dbGroupName, dbParameterGroupFamily)

    println("3. Get the parameter groups")
    describeDbParameterGroups(dbGroupName)

    println("4. Get the parameters in the group")
    describeDbParameters(dbGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBParas(dbGroupName)

    println("6. Display the updated value")
    describeDbParameters(dbGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedEngines(dbParameterGroupFamily)

    println("8. Get a list of micro instance classes available for the selected
engine")
    getMicroInstances()

    println("9. Create an RDS database instance that contains a MySql database and
uses the parameter group")
    val dbARN = createDatabaseInstance(dbGroupName, dbInstanceIdentifier, dbName,
username, userPassword)
    println("The ARN of the new database is $dbARN")

    println("10. Wait for DB instance to be ready")
    waitForDbInstanceReady(dbInstanceIdentifier)
```

```
    println("11. Create a snapshot of the DB instance")
    createDbSnapshot(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("12. Wait for DB snapshot to be ready")
    waitForSnapshotReady(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("13. Delete the DB instance")
    deleteDbInstance(dbInstanceIdentifier)

    println("14. Delete the parameter group")
    if (dbARN != null) {
        deleteParaGroup(dbGroupName, dbARN)
    }

    println("The Scenario has successfully completed.")
}

suspend fun deleteParaGroup(
    dbGroupName: String,
    dbARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false // Reset this value.
            didFind = false // Reset this value.
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(dbARN) == 0) {
                        println("$dbARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
```

```
                        // Went through the entire list and did not find the
 database name.
                        isDataDel = true
                    }
                    index++
                }
            }
        }

        // Delete the para group.
        val parameterGroupRequest =
            DeleteDbParameterGroupRequest {
                dbParameterGroupName = dbGroupName
            }
        rdsClient.deleteDbParameterGroup(parameterGroupRequest)
        println("$dbGroupName was deleted.")
    }
}

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

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

// Waits until the snapshot instance is available.
suspend fun waitForSnapshotReady(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
```

```
        DescribeDbSnapshotsRequest {
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    while (!snapshotReady) {
        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbSnapshots(snapshotsRequest)
            val snapshotList: List<DbSnapshot>? = response.dbSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

// Create an Amazon RDS snapshot.
suspend fun createDbSnapshot(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    val snapshotRequest =
        CreateDbSnapshotRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbSnapshot(snapshotRequest)
        print("The Snapshot id is ${response.dbSnapshot?.dbiResourceId}")
    }
}

// Waits until the database instance is available.
suspend fun waitForDbInstanceReady(dbInstanceIdentifierVal: String?) {
```

```
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }
    var endpoint = ""
    while (!instanceReady) {
        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint?.address.toString()
                        instanceReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}

// Create a database instance and return the ARN of the database.
suspend fun createDatabaseInstance(
    dbGroupNameVal: String?,
    dbInstanceIdentifierVal: String?,
    dbNameVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNameVal
            dbParameterGroupName = dbGroupNameVal
```

```
            engine = "mysql"
            dbInstanceClass = "db.t3.micro"
            engineVersion = "8.0.35"
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

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

// Get a list of micro instances.
suspend fun getMicroInstances() {
    val dbInstanceOptionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "mysql"
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeOrderableDbInstanceOptions(dbInstanceOptionsRequest)
        val orderableDBInstances = response.orderableDbInstanceOptions
        if (orderableDBInstances != null) {
            for (dbInstanceOption in orderableDBInstances) {
                println("The engine version is ${dbInstanceOption.engineVersion}")
                println("The engine description is ${dbInstanceOption.engine}")
            }
        }
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "mysql"
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        val dbEngines: List<DbEngineVersion>? = response.dbEngineVersions
```

```
        if (dbEngines != null) {
            for (dbEngine in dbEngines) {
                println("The engine version is ${dbEngine.engineVersion}")
                println("The engine description is ${dbEngine.dbEngineDescription}")
            }
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBParas(dbGroupName: String) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.Immediate
            parameterValue = "5"
        }

    val paraList: ArrayList<Parameter> = ArrayList()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            parameters = paraList
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbParameterGroup(groupRequest)
        println("The parameter group ${response.dbParameterGroupName} was
 successfully modified")
    }
}

// Retrieve parameters in the group.
suspend fun describeDbParameters(
    dbGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
            }
```

```
        } else {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
                source = "user"
            }
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameters(dbParameterGroupsRequest)
        val dbParameters: List<Parameter>? = response.parameters
        var paraName: String
        if (dbParameters != null) {
            for (para in dbParameters) {
                // Only print out information about either auto_increment_offset or
 auto_increment_increment.
                paraName = para.parameterName.toString()
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    System.out.println("*** The parameter value is
 ${para.parameterValue}")
                    System.out.println("*** The parameter data type is
 ${para.dataType}")
                    System.out.println("*** The parameter description is
 ${para.description}")
                    System.out.println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}

suspend fun describeDbParameterGroups(dbGroupName: String?) {
    val groupsRequest =
        DescribeDbParameterGroupsRequest {
            dbParameterGroupName = dbGroupName
            maxRecords = 20
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameterGroups(groupsRequest)
        val groups = response.dbParameterGroups
        if (groups != null) {
            for (group in groups) {
                println("The group name is ${group.dbParameterGroupName}")
```

```
                println("The group description is ${group.description}")
            }
        }
    }
}

// Create a parameter group.
suspend fun createDBParameterGroup(
    dbGroupName: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbParameterGroup(groupRequest)
        println("The group name is
 ${response.dbParameterGroup?.dbParameterGroupName}")
    }
}

// Returns a list of the available DB engines.
suspend fun describeDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            defaultOnly = true
            engine = "mysql"
            maxRecords = 20
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        val engines: List<DbEngineVersion>? = response.dbEngineVersions

        // Get all DbEngineVersion objects.
        if (engines != null) {
            for (engineOb in engines) {
                println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}.")
                println("The name of the database engine ${engineOb.engine}.")
```

```
                println("The version number of the database engine
 ${engineOb.engineVersion}")
            }
        }
    }
}

suspend fun getSecretValues(secretName: String?): String? {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient { region = "us-west-2" }.use { secretsClient ->
        val valueResponse = secretsClient.getSecretValue(valueRequest)
        return valueResponse.secretString
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

작업

`CreateDBInstance`

다음 코드 예시에서는 `CreateDBInstance`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createDatabaseInstance(
    dbInstanceIdentifierVal: String?,
    dbNamedbVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNamedbVal
            engine = "mysql"
            dbInstanceClass = "db.t3.micro" // Use a supported instance class
            engineVersion = "8.0.39" // Use a supported engine version
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

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

// Waits until the database instance is available.
suspend fun waitForInstanceReady(dbInstanceIdentifierVal: String?) {
    val sleepTime: Long = 20
```

```
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true
                    } else {
                        println("...$instanceReadyStr")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
        println("Database instance is available!")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateDBInstance를 참조하십시오.

`DeleteDBInstance`

다음 코드 예시에서는 `DeleteDBInstance`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteDBInstance를 참조하십시오.

`DescribeAccountAttributes`

다음 코드 예시에서는 `DescribeAccountAttributes`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getAccountAttributes() {
```

```
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeAccountAttributes(DescribeAccountAttributesRequest {})
        response.accountQuotas?.forEach { quotas ->
            val response = response.accountQuotas
            println("Name is: ${quotas.accountQuotaName}")
            println("Max value is ${quotas.max}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeAccountAttributes를 참조하십시오.

`DescribeDBInstances`

다음 코드 예시에서는 `DescribeDBInstances`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeInstances() {
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbInstances(DescribeDbInstancesRequest {})
        response.dbInstances?.forEach { instance ->
            println("Instance Identifier is ${instance.dbInstanceIdentifier}")
            println("The Engine is ${instance.engine}")
            println("Connection endpoint is ${instance.endpoint?.address}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeDBInstances를 참조하십시오.

`ModifyDBInstance`

다음 코드 예시에서는 `ModifyDBInstance`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun updateIntance(
    dbInstanceIdentifierVal: String?,
    masterUserPasswordVal: String?,
) {
    val request =
        ModifyDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            publiclyAccessible = true
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val instanceResponse = rdsClient.modifyDbInstance(request)
        println("The ARN of the modified database is
 ${instanceResponse.dbInstance?.dbInstanceArn}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ModifyDBInstance를 참조하십시오.

시나리오

Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제는 Amazon Aurora Serverless 데이터베이스의 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 전송하는 웹 애플리케이션을 생성하는 방법을 보여 줍니다.

SDK for Kotlin

Amazon RDS 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

Amazon Aurora Serverless 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 Spring REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 GitHub에서 전체 예제를 참조하세요.

이 예시에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

SDK for Kotlin을 사용한 Amazon RDS Data Service 예제

다음 코드 예제에서는 Amazon RDS Data Service와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시나리오

시나리오

Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제는 Amazon Aurora Serverless 데이터베이스의 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 전송하는 웹 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon RDS 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

Amazon Aurora Serverless 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 Spring REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 GitHub에서 전체 예제를 참조하세요.

이 예시에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

SDK for Kotlin을 사용한 Amazon Redshift 예제

다음 코드 예제에서는 Amazon Redshift와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업
- 시나리오

작업

`CreateCluster`

다음 코드 예시에서는 `CreateCluster`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

클러스터를 생성합니다.

```
suspend fun createCluster(
    clusterId: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val clusterRequest =
        CreateClusterRequest {
            clusterIdentifier = clusterId
            availabilityZone = "us-east-1a"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
            nodeType = "ra3.4xlarge"
            publiclyAccessible = true
            numberOfNodes = 2
        }

    RedshiftClient { region = "us-east-1" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.createCluster(clusterRequest)
        println("Created cluster ${clusterResponse.cluster?.clusterIdentifier}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateCluster를 참조하십시오.

`DeleteCluster`

다음 코드 예시에서는 `DeleteCluster`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

클러스터를 삭제합니다.

```
suspend fun deleteRedshiftCluster(clusterId: String?) {
    val request =
        DeleteClusterRequest {
            clusterIdentifier = clusterId
            skipFinalClusterSnapshot = true
        }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val response = redshiftClient.deleteCluster(request)
        println("The status is ${response.cluster?.clusterStatus}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteCluster를 참조하십시오.

`DescribeClusters`

다음 코드 예시에서는 `DescribeClusters`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

클러스터를 설명하세요.

```
suspend fun describeRedshiftClusters() {
```

```
    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse =
 redshiftClient.describeClusters(DescribeClustersRequest {})
        val clusterList = clusterResponse.clusters

        if (clusterList != null) {
            for (cluster in clusterList) {
                println("Cluster database name is ${cluster.dbName}")
                println("Cluster status is ${cluster.clusterStatus}")
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeClusters를 참조하십시오.

`ModifyCluster`

다음 코드 예시에서는 `ModifyCluster`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

클러스터를 수정합니다.

```
suspend fun modifyCluster(clusterId: String?) {
    val modifyClusterRequest =
        ModifyClusterRequest {
            clusterIdentifier = clusterId
            preferredMaintenanceWindow = "wed:07:30-wed:08:00"
        }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.modifyCluster(modifyClusterRequest)
        println(
```

```
            "The modified cluster was successfully modified and has
 ${clusterResponse.cluster?.preferredMaintenanceWindow} as the maintenance window",
        )
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ModifyCluster를 참조하십시오.

시나리오

Amazon Redshift 데이터를 추적하는 웹 애플리케이션 만들기

다음 코드 예제는 Amazon Redshift 데이터베이스를 사용하여 작업 항목을 추적하고 보고하는 웹 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon Redshift 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

Amazon Redshift 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 Spring REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 GitHub에서 전체 예제를 참조하세요.

이 예시에서 사용되는 서비스

- Amazon Redshift
- Amazon SES

SDK for Kotlin을 사용한 Amazon Rekognition 예제

다음 코드 예제에서는 Amazon Rekognition과 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업
- 시나리오

작업

`CompareFaces`

다음 코드 예시에서는 `CompareFaces`을 사용하는 방법을 보여 줍니다.

자세한 내용은 이미지에 있는 얼굴 비교를 참조하십시오.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun compareTwoFaces(
    similarityThresholdVal: Float,
    sourceImageVal: String,
    targetImageVal: String,
) {
    val sourceBytes = (File(sourceImageVal).readBytes())
    val targetBytes = (File(targetImageVal).readBytes())

    // Create an Image object for the source image.
    val souImage =
        Image {
            bytes = sourceBytes
        }

    val tarImage =
        Image {
            bytes = targetBytes
```

```
        }

    val facesRequest =
        CompareFacesRequest {
            sourceImage = souImage
            targetImage = tarImage
            similarityThreshold = similarityThresholdVal
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->

        val compareFacesResult = rekClient.compareFaces(facesRequest)
        val faceDetails = compareFacesResult.faceMatches

        if (faceDetails != null) {
            for (match: CompareFacesMatch in faceDetails) {
                val face = match.face
                val position = face?.boundingBox
                if (position != null) {
                    println("Face at ${position.left} ${position.top} matches with
 ${face.confidence} % confidence.")
                }
            }
        }

        val uncompared = compareFacesResult.unmatchedFaces
        if (uncompared != null) {
            println("There was ${uncompared.size} face(s) that did not match")
        }

        println("Source image rotation:
 ${compareFacesResult.sourceImageOrientationCorrection}")
        println("target image rotation:
 ${compareFacesResult.targetImageOrientationCorrection}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CompareFaces를 참조하세요.

`CreateCollection`

다음 코드 예시에서는 `CreateCollection`을 사용하는 방법을 보여 줍니다.

자세한 내용은 컬렉션 생성을 참조하십시오.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createMyCollection(collectionIdVal: String) {
    val request =
        CreateCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateCollection을 참조하세요.

`DeleteCollection`

다음 코드 예시에서는 `DeleteCollection`을 사용하는 방법을 보여 줍니다.

자세한 내용은 컬렉션 삭제를 참조하세요.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteMyCollection(collectionIdVal: String) {
    val request =
        DeleteCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteCollection을 참조하세요.

`DeleteFaces`

다음 코드 예시에서는 `DeleteFaces`을 사용하는 방법을 보여 줍니다.

자세한 내용은 컬렉션에서 얼굴 삭제를 참조하세요.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteFacesCollection(
    collectionIdVal: String?,
    faceIdVal: String,
) {
    val deleteFacesRequest =
        DeleteFacesRequest {
            collectionId = collectionIdVal
            faceIds = listOf(faceIdVal)
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
```

```
        println("$faceIdVal was deleted from the collection")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteFaces를 참조하세요.

`DescribeCollection`

다음 코드 예시에서는 `DescribeCollection`을 사용하는 방법을 보여 줍니다.

자세한 내용은 컬렉션 설명을 참조하십시오.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeColl(collectionName: String) {
    val request =
        DescribeCollectionRequest {
            collectionId = collectionName
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.describeCollection(request)
        println("The collection Arn is ${response.collectionArn}")
        println("The collection contains this many faces ${response.faceCount}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeCollection을 참조하세요.

`DetectFaces`

다음 코드 예시에서는 `DetectFaces`을 사용하는 방법을 보여 줍니다.

자세한 내용은 이미지에서 얼굴 감지를 참조하세요.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun detectFacesinImage(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectFacesRequest {
            attributes = listOf(Attribute.All)
            image = souImage
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectFaces(request)
        response.faceDetails?.forEach { face ->
            val ageRange = face.ageRange
            println("The detected face is estimated to be between ${ageRange?.low}
 and ${ageRange?.high} years old.")
            println("There is a smile ${face.smile?.value}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DetectFaces를 참조하세요.

`DetectLabels`

다음 코드 예시에서는 `DetectLabels`을 사용하는 방법을 보여 줍니다.

자세한 내용은 이미지에서 레이블 감지를 참조하십시오.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun detectImageLabels(sourceImage: String) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }
    val request =
        DetectLabelsRequest {
            image = souImage
            maxLabels = 10
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectLabels(request)
        response.labels?.forEach { label ->
            println("${label.name} : ${label.confidence}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DetectLabels를 참조하세요.

`DetectModerationLabels`

다음 코드 예시에서는 `DetectModerationLabels`을 사용하는 방법을 보여 줍니다.

자세한 내용은 부적절한 이미지 감지를 참조하십시오.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun detectModLabels(sourceImage: String) {
    val myImage =
        Image {
            this.bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectModerationLabelsRequest {
            image = myImage
            minConfidence = 60f
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectModerationLabels(request)
        response.moderationLabels?.forEach { label ->
            println("Label: ${label.name} - Confidence: ${label.confidence} %
 Parent: ${label.parentName}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DetectModerationLabels를 참조하세요.

`DetectText`

다음 코드 예시에서는 `DetectText`을 사용하는 방법을 보여 줍니다.

자세한 내용은 이미지에서 텍스트 감지를 참조하십시오.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun detectTextLabels(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectTextRequest {
            image = souImage
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectText(request)
        response.textDetections?.forEach { text ->
            println("Detected: ${text.detectedText}")
            println("Confidence: ${text.confidence}")
            println("Id: ${text.id}")
            println("Parent Id:  ${text.parentId}")
            println("Type: ${text.type}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DetectText를 참조하세요.

`IndexFaces`

다음 코드 예시에서는 `IndexFaces`을 사용하는 방법을 보여 줍니다.

자세한 내용은 컬렉션에 얼굴 추가를 참조하세요.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun addToCollection(
    collectionIdVal: String?,
    sourceImage: String,
) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        IndexFacesRequest {
            collectionId = collectionIdVal
            image = souImage
            maxFaces = 1
            qualityFilter = QualityFilter.Auto
            detectionAttributes = listOf(Attribute.Default)
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val facesResponse = rekClient.indexFaces(request)

        // Display the results.
        println("Results for the image")
        println("\n Faces indexed:")
        facesResponse.faceRecords?.forEach { faceRecord ->
            println("Face ID: ${faceRecord.face?.faceId}")
            println("Location: ${faceRecord.faceDetail?.boundingBox}")
        }

        println("Faces not indexed:")
        facesResponse.unindexedFaces?.forEach { unindexedFace ->
            println("Location: ${unindexedFace.faceDetail?.boundingBox}")
            println("Reasons:")
```

```
            unindexedFace.reasons?.forEach { reason ->
                println("Reason:  $reason")
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 IndexFaces를 참조하세요.

`ListCollections`

다음 코드 예시에서는 `ListCollections`을 사용하는 방법을 보여 줍니다.

자세한 내용은 컬렉션 나열을 참조하십시오.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listAllCollections() {
    val request =
        ListCollectionsRequest {
            maxResults = 10
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListCollections를 참조하세요.

`ListFaces`

다음 코드 예시에서는 `ListFaces`을 사용하는 방법을 보여 줍니다.

자세한 내용은 컬렉션 내 얼굴 나열을 참조하세요.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listFacesCollection(collectionIdVal: String?) {
    val request =
        ListFacesRequest {
            collectionId = collectionIdVal
            maxResults = 10
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listFaces(request)
        response.faces?.forEach { face ->
            println("Confidence level there is a face: ${face.confidence}")
            println("The face Id value is ${face.faceId}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListFaces를 참조하세요.

`RecognizeCelebrities`

다음 코드 예시에서는 `RecognizeCelebrities`을 사용하는 방법을 보여 줍니다.

자세한 내용은 이미지에서 유명인 인식을 참조하세요.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        RecognizeCelebritiesRequest {
            image = souImage
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.recognizeCelebrities(request)
        response.celebrityFaces?.forEach { celebrity ->
            println("Celebrity recognized: ${celebrity.name}")
            println("Celebrity ID:${celebrity.id}")
            println("Further information (if available):")
            celebrity.urls?.forEach { url ->
                println(url)
            }
        }
        println("${response.unrecognizedFaces?.size} face(s) were unrecognized.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 RecognizeCelebrities를 참조하십시오.

시나리오

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예시에서 사용되는 서비스

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

동영상 내 정보 감지

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Amazon Rekognition 작업을 시작하여 동영상에서 사람, 사물, 텍스트와 같은 요소를 탐지하세요.
- 작업이 완료될 때까지 작업 상태를 확인하세요.
- 각 작업에서 감지한 요소의 목록을 출력합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 버킷에 저장된 동영상에서 얼굴을 감지합니다.

```
suspend fun startFaceDetection(
    channelVal: NotificationChannel?,
    bucketVal: String,
    videoVal: String,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }

    val request =
        StartFaceDetectionRequest {
            jobTag = "Faces"
            faceAttributes = FaceAttributes.All
            notificationChannel = channelVal
            video = vidOb
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val startLabelDetectionResult = rekClient.startFaceDetection(request)
        startJobId = startLabelDetectionResult.jobId.toString()
    }
}

suspend fun getFaceResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var response: GetFaceDetectionResponse? = null

        val recognitionRequest =
            GetFaceDetectionRequest {
                jobId = startJobId
                maxResults = 10
            }
```

```
        // Wait until the job succeeds.
        while (!finished) {
            response = rekClient.getFaceDetection(recognitionRequest)
            status = response.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = response?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        // Show face information.
        response?.faces?.forEach { face ->
            println("Age: ${face.face?.ageRange}")
            println("Face: ${face.face?.beard}")
            println("Eye glasses: ${face?.face?.eyeglasses}")
            println("Mustache: ${face.face?.mustache}")
            println("Smile: ${face.face?.smile}")
        }
    }
}
```

Amazon S3 버킷에 저장된 동영상에서 부적절하거나 불쾌감을 주는 콘텐츠를 감지합니다.

```
suspend fun startModerationDetection(
    channel: NotificationChannel?,
    bucketVal: String?,
    videoVal: String?,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
```

```
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }
    val request =
        StartContentModerationRequest {
            jobTag = "Moderation"
            notificationChannel = channel
            video = vidOb
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val startModDetectionResult = rekClient.startContentModeration(request)
        startJobId = startModDetectionResult.jobId.toString()
    }
}

suspend fun getModResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var modDetectionResponse: GetContentModerationResponse? = null

        val modRequest =
            GetContentModerationRequest {
                jobId = startJobId
                maxResults = 10
            }

        // Wait until the job succeeds.
        while (!finished) {
            modDetectionResponse = rekClient.getContentModeration(modRequest)
            status = modDetectionResponse.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }
```

```
        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = modDetectionResponse?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        modDetectionResponse?.moderationLabels?.forEach { mod ->
            val seconds: Long = mod.timestamp / 1000
            print("Mod label: $seconds ")
            println(mod.moderationLabel)
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection
  - StartTextDetection

이미지에서 객체 감지

다음 코드 예제는 Amazon Rekognition을 사용하여 이미지의 범주별로 객체를 감지하는 앱을 구축하는 방법을 보여줍니다.

SDK for Kotlin

Amazon Rekognition Kotlin API를 사용하여 Amazon Simple Storage Service (Amazon S3) 버킷에 있는 이미지에서 범주별로 객체를 식별하기 위해 Amazon Rekognition을 사용하여 앱을 만드는 방법을 보여줍니다. 이 앱은 Amazon Simple Email Service(Amazon SES)를 사용하여 결과와 함께 이메일 알림을 관리자에게 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하십시오.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES

SDK for Kotlin을 사용한 Route 53 도메인 등록 예제

다음 코드 예제에서는 Route 53 도메인 등록과 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시작

Route 53 도메인 등록 소개

다음 코드 예제는 Route 53 도메인 등록 사용을 시작하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
           <domainType>

       Where:
           domainType - The domain type (for example, com).
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val domainType = args[0]
    println("Invokes ListPrices using a Paginated method.")
    listPricesPaginated(domainType)
}

suspend fun listPricesPaginated(domainType: String) {
    val pricesRequest =
        ListPricesRequest {
            maxItems = 10
            tld = domainType
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
```

```
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}
```

- API 세부 정보는AWS SDK for Kotlin API 참조의 ListPrices를 참조하십시오.

주제

- 기본 사항
- 작업

기본 사항

기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 현재 도메인과 작년의 작업을 나열합니다.
- 작년의 결제 내역과 도메인 유형의 가격을 봅니다.
- 도메인 제안을 가져옵니다.
- 도메인 가용성 및 이전 가능성을 확인합니다.
- 선택 사항으로 도메인 등록을 요청할 수도 있습니다.
- 작업 세부 정보를 가져옵니다.
- 선택 사항으로 도메인 세부 정보를 가져올 수 있습니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin code example performs the following operations:

1. List current domains.
2. List operations in the past year.
3. View billing for the account in the past year.
4. View prices for domain types.
5. Get domain suggestions.
6. Check domain availability.
7. Check domain transferability.
8. Request a domain registration.
9. Get operation details.
10. Optionally, get domain details.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <domainType> <phoneNumber> <email> <domainSuggestion> <firstName>
 <lastName> <city>
        Where:
           domainType - The domain type (for example, com).
           phoneNumber - The phone number to use (for example, +1.2065550100)
           email - The email address to use.
           domainSuggestion - The domain suggestion (for example, findmy.example).
           firstName - The first name to use to register a domain.
           lastName -  The last name to use to register a domain.
           city - The city to use to register a domain.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }
```

```
    val domainType = args[0]
    val phoneNumber = args[1]
    val email = args[2]
    val domainSuggestion = args[3]
    val firstName = args[4]
    val lastName = args[5]
    val city = args[6]

    println(DASHES)
    println("Welcome to the Amazon Route 53 domains example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List current domains.")
    listDomains()
    println(DASHES)

    println(DASHES)
    println("2. List operations in the past year.")
    listOperations()
    println(DASHES)

    println(DASHES)
    println("3. View billing for the account in the past year.")
    listBillingRecords()
    println(DASHES)

    println(DASHES)
    println("4. View prices for domain types.")
    listAllPrices(domainType)
    println(DASHES)

    println(DASHES)
    println("5. Get domain suggestions.")
    listDomainSuggestions(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("6. Check domain availability.")
    checkDomainAvailability(domainSuggestion)
    println(DASHES)

    println(DASHES)
```

```
    println("7. Check domain transferability.")
    checkDomainTransferability(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("8. Request a domain registration.")
    val opId = requestDomainRegistration(domainSuggestion, phoneNumber, email,
 firstName, lastName, city)
    println(DASHES)

    println(DASHES)
    println("9. Get operation details.")
    getOperationalDetail(opId)
    println(DASHES)

    println(DASHES)
    println("10. Get domain details.")
    println("Note: You must have a registered domain to get details.")
    println("Otherwise an exception is thrown that states ")
    println("Domain xxxxxxx not found in xxxxxxx account.")
    getDomainDetails(domainSuggestion)
    println(DASHES)
}

suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getDomainDetail(detailRequest)
        println("The contact first name is
 ${response.registrantContact?.firstName}")
        println("The contact last name is ${response.registrantContact?.lastName}")
        println("The contact org name is
 ${response.registrantContact?.organizationName}")
    }
}

suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
```

```
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}

suspend fun requestDomainRegistration(
    domainSuggestion: String?,
    phoneNumberVal: String?,
    emailVal: String?,
    firstNameVal: String?,
    lastNameVal: String?,
    cityVal: String?,
): String? {
    val contactDetail =
        ContactDetail {
            contactType = ContactType.Company
            state = "LA"
            countryCode = CountryCode.In
            email = emailVal
            firstName = firstNameVal
            lastName = lastNameVal
            city = cityVal
            phoneNumber = phoneNumberVal
            organizationName = "My Org"
            addressLine1 = "My Address"
            zipCode = "123 123"
        }

    val domainRequest =
        RegisterDomainRequest {
            adminContact = contactDetail
            registrantContact = contactDetail
            techContact = contactDetail
            domainName = domainSuggestion
            autoRenew = true
            durationInYears = 1
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
```

```
}

suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}

suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
}

suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getDomainSuggestions(suggestionsRequest)
        response.suggestionsList?.forEach { suggestion ->
            println("Suggestion Name: ${suggestion.domainName}")
            println("Availability: ${suggestion.availability}")
            println(" ")
        }
    }
}

suspend fun listAllPrices(domainType: String?) {
```

```
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}

suspend fun listBillingRecords() {
    val currentDate = Date()
    val localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    val localDateTime2 = localDateTime.minusYears(1)
    val myStartTime = localDateTime2.toInstant(zoneOffset)
    val myEndTime = localDateTime.toInstant(zoneOffset)
    val timeStart: Instant? = myStartTime?.let { Instant(it) }
    val timeEnd: Instant? = myEndTime?.let { Instant(it) }

    val viewBillingRequest =
        ViewBillingRequest {
            start = timeStart
            end = timeEnd
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .viewBillingPaginated(viewBillingRequest)
            .transform { it.billingRecords?.forEach { obj -> emit(obj) } }
            .collect { billing ->
```

```
                println("Bill Date: ${billing.billDate}")
                println("Operation: ${billing.operation}")
                println("Price: ${billing.price}")
            }
    }
}

suspend fun listOperations() {
    val currentDate = Date()
    var localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    localDateTime = localDateTime.minusYears(1)
    val myTime: java.time.Instant? = localDateTime.toInstant(zoneOffset)
    val time2: Instant? = myTime?.let { Instant(it) }
    val operationsRequest =
        ListOperationsRequest {
            submittedSince = time2
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listOperationsPaginated(operationsRequest)
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("Operation Id: ${content.operationId}")
                println("Status: ${content.status}")
                println("Date: ${content.submittedDate}")
            }
    }
}

suspend fun listDomains() {
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listDomainsPaginated(ListDomainsRequest {})
            .transform { it.domains?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("The domain name is ${content.domainName}")
            }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

작업

`CheckDomainAvailability`

다음 코드 예시에서는 `CheckDomainAvailability`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CheckDomainAvailability를 참조하십시오.

`CheckDomainTransferability`

다음 코드 예시에서는 `CheckDomainTransferability`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CheckDomainTransferability를 참조하십시오.

`GetDomainDetail`

다음 코드 예시에서는 `GetDomainDetail`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getDomainDetail(detailRequest)
        println("The contact first name is
 ${response.registrantContact?.firstName}")
        println("The contact last name is ${response.registrantContact?.lastName}")
        println("The contact org name is
 ${response.registrantContact?.organizationName}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetDomainDetail을 참조하십시오.

`GetDomainSuggestions`

다음 코드 예시에서는 `GetDomainSuggestions`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listDomainSuggestions(domainSuggestion: String?) {
```

```
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getDomainSuggestions(suggestionsRequest)
        response.suggestionsList?.forEach { suggestion ->
            println("Suggestion Name: ${suggestion.domainName}")
            println("Availability: ${suggestion.availability}")
            println(" ")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetDomainSuggestions를 참조하십시오.

`GetOperationDetail`

다음 코드 예시에서는 `GetOperationDetail`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetOperationDetail을 참조하십시오.

`ListDomains`

다음 코드 예시에서는 `ListDomains`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listDomains() {
    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listDomainsPaginated(ListDomainsRequest {})
            .transform { it.domains?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("The domain name is ${content.domainName}")
            }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListDomains를 참조하십시오.

`ListOperations`

다음 코드 예시에서는 `ListOperations`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listOperations() {
    val currentDate = Date()
    var localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    localDateTime = localDateTime.minusYears(1)
    val myTime: java.time.Instant? = localDateTime.toInstant(zoneOffset)
    val time2: Instant? = myTime?.let { Instant(it) }
    val operationsRequest =
        ListOperationsRequest {
            submittedSince = time2
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listOperationsPaginated(operationsRequest)
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("Operation Id: ${content.operationId}")
                println("Status: ${content.status}")
                println("Date: ${content.submittedDate}")
            }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListOperations를 참조하십시오.

`ListPrices`

다음 코드 예시에서는 `ListPrices`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listAllPrices(domainType: String?) {
```

```
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}
```

- API 세부 정보는AWS SDK for Kotlin API 참조의 ListPrices를 참조하십시오.

`RegisterDomain`

다음 코드 예시에서는 `RegisterDomain`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun requestDomainRegistration(
    domainSuggestion: String?,
    phoneNumberVal: String?,
    emailVal: String?,
```

```
    firstNameVal: String?,
    lastNameVal: String?,
    cityVal: String?,
): String? {
    val contactDetail =
        ContactDetail {
            contactType = ContactType.Company
            state = "LA"
            countryCode = CountryCode.In
            email = emailVal
            firstName = firstNameVal
            lastName = lastNameVal
            city = cityVal
            phoneNumber = phoneNumberVal
            organizationName = "My Org"
            addressLine1 = "My Address"
            zipCode = "123 123"
        }

    val domainRequest =
        RegisterDomainRequest {
            adminContact = contactDetail
            registrantContact = contactDetail
            techContact = contactDetail
            domainName = domainSuggestion
            autoRenew = true
            durationInYears = 1
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 RegisterDomain을 참조하십시오.

`ViewBilling`

다음 코드 예시에서는 `ViewBilling`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listBillingRecords() {
    val currentDate = Date()
    val localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    val localDateTime2 = localDateTime.minusYears(1)
    val myStartTime = localDateTime2.toInstant(zoneOffset)
    val myEndTime = localDateTime.toInstant(zoneOffset)
    val timeStart: Instant? = myStartTime?.let { Instant(it) }
    val timeEnd: Instant? = myEndTime?.let { Instant(it) }

    val viewBillingRequest =
        ViewBillingRequest {
            start = timeStart
            end = timeEnd
        }

    Route53DomainsClient { region = "us-east-1" }.use { route53DomainsClient ->
        route53DomainsClient
            .viewBillingPaginated(viewBillingRequest)
            .transform { it.billingRecords?.forEach { obj -> emit(obj) } }
            .collect { billing ->
                println("Bill Date: ${billing.billDate}")
                println("Operation: ${billing.operation}")
                println("Price: ${billing.price}")
            }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ViewBilling을 참조하십시오.

SDK for Kotlin을 사용한 Amazon S3 예제

다음 코드 예제에서는 Amazon S3에서 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 전체 소스 코드에 대한 링크가 포함되어 있습니다.

주제

- 기본 사항
- 작업
- 시나리오

기본 사항

기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 버킷을 만들고 버킷에 파일을 업로드합니다.
- 버킷에서 객체를 다운로드합니다.
- 버킷의 하위 폴더에 객체를 복사합니다.
- 버킷의 객체를 나열합니다.
- 버킷 객체와 버킷을 삭제합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <bucketName> <key> <objectPath> <savePath> <toBucket>

    Where:
        bucketName - The Amazon S3 bucket to create.
        key - The key to use.
        objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf).
        savePath - The path where the file is saved after it's downloaded (for
 example, C:/AWS/book2.pdf).
        toBucket - An Amazon S3 bucket to where an object is copied to (for example,
 C:/AWS/book2.pdf).
        """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val bucketName = args[0]
    val key = args[1]
    val objectPath = args[2]
    val savePath = args[3]
    val toBucket = args[4]

    // Create an Amazon S3 bucket.
    createBucket(bucketName)

    // Update a local file to the Amazon S3 bucket.
    putObject(bucketName, key, objectPath)

    // Download the object to another local file.
```

```
    getObjectFromMrap(bucketName, key, savePath)

    // List all objects located in the Amazon S3 bucket.
    listBucketObs(bucketName)

    // Copy the object to another Amazon S3 bucket
    copyBucketOb(bucketName, key, toBucket)

    // Delete the object from the Amazon S3 bucket.
    deleteBucketObs(bucketName, key)

    // Delete the Amazon S3 bucket.
    deleteBucket(bucketName)
    println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            this.body = Paths.get(objectPath).asByteStream()
        }
```

```
    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObjectFromMrap(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucketObs(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->

        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The owner is ${myObject.owner}")
        }
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
```

```
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
```

```
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

작업

`CopyObject`

다음 코드 예시에서는 `CopyObject`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CopyObject를 참조하십시오.

`CreateBucket`

다음 코드 예시에서는 `CreateBucket`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createNewBucket(bucketName: String) {
    val request =
```

```
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateBucket를 참조하세요.

`CreateMultiRegionAccessPoint`

다음 코드 예시에서는 `CreateMultiRegionAccessPoint`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

us-west-2 리전으로 요청을 보내도록 S3 제어 클라이언트를 구성합니다.

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

다중 리전 액세스 포인트를 생성합니다.

```
    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"
    }
```

다중 리전 액세스 포인트가 사용 가능해질 때까지 기다립니다.

```
        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)
            }
        }
```

- 자세한 내용은 AWS SDK for Kotlin 개발자 안내서를 참조하십시오.
- API 세부 정보는 Kotlin API 참조용AWS SDK의 CreateMultiRegionAccessPoint를 참조하세요.

`DeleteBucketPolicy`

다음 코드 예시에서는 `DeleteBucketPolicy`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteBucketPolicy를 참조하십시오.

`DeleteObjects`

다음 코드 예시에서는 `DeleteObjects`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteBucketObjects(
    bucketName: String,
    objectName: String,
) {
    val objectId =
```

```
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteObjects를 참조하십시오.

`GetBucketPolicy`

다음 코드 예시에서는 `GetBucketPolicy`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
```

```
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetBucketPolicy를 참조하십시오.

`GetObject`

다음 코드 예시에서는 `GetObject`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getObjectBytes(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetObject를 참조하십시오.

`GetObjectAcl`

다음 코드 예시에서는 `GetObjectAcl`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetObjectAcl를 참조하십시오.

`ListObjectsV2`

다음 코드 예시에서는 `ListObjectsV2`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${myObject.size?.let { calKb(it) }} KBs")
            println("The owner is ${myObject.owner}")
        }
    }
}

private fun calKb(intValue: Long): Long = intValue / 1024
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListObjectsV2를 참조하십시오.

`PutBucketAcl`

다음 코드 예시에서는 `PutBucketAcl`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun setBucketAcl(
    bucketName: String,
    idVal: String,
) {
    val myGrant =
        Grantee {
            id = idVal
            type = Type.CanonicalUser
        }

    val ownerGrant =
        Grant {
            grantee = myGrant
            permission = Permission.FullControl
        }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb =
        Owner {
            id = idVal
        }

    val acl =
        AccessControlPolicy {
            owner = ownerOb
            grants = grantList
        }

    val request =
        PutBucketAclRequest {
            bucket = bucketName
            accessControlPolicy = acl
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 PutBucketAcl를 참조하십시오.

`PutObject`

다음 코드 예시에서는 `PutObject`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            body = File(objectPath).asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 PutObject를 참조하십시오.

시나리오

미리 서명된 URL 생성

다음 코드 예제에서는 Amazon S3에 대해 미리 서명된 URL을 생성하고 객체를 업로드하는 방법을 보여줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

`GetObject` 미리 서명된 GetObject 요청을 만들고 URL을 사용하여 객체를 다운로드합니다.

```
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
}
```

고급 옵션을 사용하여 `GetObject` 미리 서명된 요청을 생성합니다.

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): HttpRequest {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
            signingDate = Instant.now() + 12.hours // Presigned request can be used
 12 hours from now.
            algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
            signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
            expiresAfter = 8.hours // Presigned request expires 8 hours later.
        }
    return presignedRequest
}
```

PutObject 미리 서명된 요청을 만들고 이를 사용하여 객체를 업로드합니다.

```
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)
}
```

- 자세한 내용은 AWS SDK for Kotlin 개발자 안내서를 참조하십시오.

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예시에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition

- Amazon S3
- Amazon SNS

이미지에서 객체 감지

다음 코드 예제는 Amazon Rekognition을 사용하여 이미지의 범주별로 객체를 감지하는 앱을 구축하는 방법을 보여줍니다.

SDK for Kotlin

Amazon Rekognition Kotlin API를 사용하여 Amazon Simple Storage Service (Amazon S3) 버킷에 있는 이미지에서 범주별로 객체를 식별하기 위해 Amazon Rekognition을 사용하여 앱을 만드는 방법을 보여줍니다. 이 앱은 Amazon Simple Email Service(Amazon SES)를 사용하여 결과와 함께 이메일 알림을 관리자에게 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하십시오.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES

다중 리전 액세스 포인트에서 객체 생성

다음 코드 예제에서는 다중 리전 액세스 포인트에서 객체를 가져오는 방법을 보여줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

비대칭 Sigv4(Sigv4a) 서명 알고리즘을 사용하도록 S3 클라이언트를 구성합니다.

```
        suspend fun createS3Client(): S3Client {
```

```
            // Configure your S3Client to use the Asymmetric Sigv4 (Sigv4a) signing
algorithm.
            val sigV4AScheme = SigV4AsymmetricAuthScheme(CrtAwsSigner)
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4AScheme)
            }
            return s3
        }
```

버킷 이름 대신 다중 리전 액세스 포인트 ARN을 사용하영 객체를 가져옵니다.

```
    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
    }
```

- 자세한 내용은 AWS SDK for Kotlin 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetObject를 참조하십시오.

SDK for Kotlin을 사용한 SageMaker AI 예제

다음 코드 예제에서는 AWS SDK for Kotlin을 SageMaker AI와 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시작

Hello SageMaker AI

다음 코드 예제에서는 SageMaker AI 사용을 시작하는 방법을 보여줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listBooks() {
    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
 sageMakerClient.listNotebookInstances(ListNotebookInstancesRequest {})
        response.notebookInstances?.forEach { item ->
            println("The notebook name is: ${item.notebookInstanceName}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListNotebookInstances를 참조하십시오.

주제

- 작업
- 시나리오

작업

`CreatePipeline`

다음 코드 예시에서는 `CreatePipeline`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Create a pipeline from the example pipeline JSON.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
 String?, pipelineNameVal: String?) {
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
        println(jsonObject)

        // Create the pipeline.
        val pipelineRequest = CreatePipelineRequest {
            pipelineDescription = "Kotlin SDK example pipeline"
            roleArn = roleArnVal
            pipelineName = pipelineNameVal
            pipelineDefinition = jsonObject.toString()
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
```

```
            sageMakerClient.createPipeline(pipelineRequest)
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreatePipeline을 참조하세요.

`DeletePipeline`

다음 코드 예시에서는 `DeletePipeline`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Delete a SageMaker pipeline by name.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.deletePipeline(pipelineRequest)
        println("*** Successfully deleted $pipelineNameVal")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeletePipeline을 참조하십시오.

`DescribePipelineExecution`

다음 코드 예시에서는 `DescribePipelineExecution`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
 sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribePipelineExecution을 참조하십시오.

`StartPipelineExecution`

다음 코드 예시에서는 `StartPipelineExecution`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Start a pipeline run with job configurations.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn: String?,
 pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
        value = roleArn
    }
    val para2 = Parameter {
        name = "parameter_queue_url"
        value = queueUrl
    }

    val inputJSON = """{
        "DataSourceConfig": {
        "S3Data": {
            "S3Uri": "s3://$bucketName/samplefiles/latlongtest.csv"
        },
        "Type": "S3_DATA"
        },
        "DocumentType": "CSV"
    }"""
    println(inputJSON)
```

```
    val para3 = Parameter {
        name = "parameter_vej_input_config"
        value = inputJSON
    }

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    val jobS3Data = VectorEnrichmentJobS3Data {
        s3Uri = output
    }

    val outputConfig = ExportVectorEnrichmentJobOutputConfig {
        s3Data = jobS3Data
    }

    val gson4: String = gson.toJson(outputConfig)
    val para4: Parameter = Parameter {
        name = "parameter_vej_export_config"
        value = gson4
    }
    println("parameter_vej_export_config:" + gson.toJson(outputConfig))

    val para5JSON =
        "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName\":
\"Longitude\",\"YAttributeName\":\"Latitude\"}}"

    val para5: Parameter = Parameter {
        name = "parameter_step_1_vej_config"
        value = para5JSON
    }

    parameters.add(para1)
    parameters.add(para2)
    parameters.add(para3)
    parameters.add(para4)
    parameters.add(para5)

    val pipelineExecutionRequest = StartPipelineExecutionRequest {
        pipelineExecutionDescription = "Created using Kotlin SDK"
        pipelineExecutionDisplayName = "$pipelineName-example-execution"
        pipelineParameters = parameters
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
```

```
        val response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 StartPipelineExecution를 참조하십시오.

시나리오

지리공간 작업 및 파이프라인으로 시작하기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 파이프라인의 리소스를 설정하세요.
- 지리 공간 작업을 실행하는 파이프라인을 설정합니다.
- 파이프라인 실행을 시작합니다.
- 실행 상태를 모니터링합니다.
- 파이프라인의 출력을 볼 수 있습니다.
- 리소스를 정리합니다.

자세한 내용은 Community. AWS SDKs를 사용하여 SageMaker 파이프라인 생성 및 실행을 참조하세요.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
val DASHES = String(CharArray(80)).replace("\u0000", "-")
private var eventSourceMapping = ""

suspend fun main(args: Array<String>) {
    val usage = """
```

```
    Usage:
          <sageMakerRoleName> <lambdaRoleName> <functionName> <functionKey>
 <queueName> <bucketName> <bucketFunction> <lnglatData> <spatialPipelinePath>
 <pipelineName>

    Where:
        sageMakerRoleName - The name of the Amazon SageMaker role.
        lambdaRoleName - The name of the AWS Lambda role.
        functionName - The name of the AWS Lambda function (for
 example,SageMakerExampleFunction).
        functionKey - The name of the Amazon S3 key name that represents the Lambda
 function (for example, SageMakerLambda.zip).
        queueName - The name of the Amazon Simple Queue Service (Amazon SQS) queue.
        bucketName - The name of the Amazon Simple Storage Service (Amazon S3)
 bucket.
        bucketFunction - The name of the Amazon S3 bucket that contains the Lambda
 ZIP file.
        lnglatData - The file location of the latlongtest.csv file required for this
 use case.
        spatialPipelinePath - The file location of the GeoSpatialPipeline.json file
 required for this use case.
        pipelineName - The name of the pipeline to create (for example, sagemaker-
sdk-example-pipeline).
    """

    if (args.size != 10) {
        println(usage)
        exitProcess(1)
    }

    val sageMakerRoleName = args[0]
    val lambdaRoleName = args[1]
    val functionKey = args[2]
    val functionName = args[3]
    val queueName = args[4]
    val bucketName = args[5]
    val bucketFunction = args[6]
    val lnglatData = args[7]
    val spatialPipelinePath = args[8]
    val pipelineName = args[9]
    val handlerName = "org.example.SageMakerLambdaFunction::handleRequest"

    println(DASHES)
    println("Welcome to the Amazon SageMaker pipeline example scenario.")
```

```
    println(
        """
         This example workflow will guide you through setting up and running an
         Amazon SageMaker pipeline. The pipeline uses an AWS Lambda function and an
         Amazon SQS Queue. It runs a vector enrichment reverse geocode job to
         reverse geocode addresses in an input file and store the results in an
export file.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println("First, we will set up the roles, functions, and queue needed by the
SageMaker pipeline.")
    val lambdaRoleArn: String = checkLambdaRole(lambdaRoleName)
    val sageMakerRoleArn: String = checkSageMakerRole(sageMakerRoleName)
    val functionArn = checkFunction(functionName, bucketFunction, functionKey,
handlerName, lambdaRoleArn)
    val queueUrl = checkQueue(queueName, functionName)
    println(DASHES)

    println(DASHES)
    println("Setting up bucket $bucketName")
    if (!checkBucket(bucketName)) {
        setupBucket(bucketName)
        println("Put $lnglatData into $bucketName")
        val objectKey = "samplefiles/latlongtest.csv"
        putS3Object(bucketName, objectKey, lnglatData)
    }
    println(DASHES)

    println(DASHES)
    println("Now we can create and run our pipeline.")
    setupPipeline(spatialPipelinePath, sageMakerRoleArn, functionArn, pipelineName)
    val pipelineExecutionARN = executePipeline(bucketName, queueUrl,
sageMakerRoleArn, pipelineName)
    println("The pipeline execution ARN value is $pipelineExecutionARN")
    waitForPipelineExecution(pipelineExecutionARN)
    println("Wait 30 secs to get output results $bucketName")
    TimeUnit.SECONDS.sleep(30)
    getOutputResults(bucketName)
    println(DASHES)

    println(DASHES)
```

```
    println(
        """
            The pipeline has completed. To view the pipeline and runs in SageMaker
 Studio, follow these instructions:
            https://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-studio.html
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println("Do you want to delete the AWS resources used in this Workflow? (y/n)")
    val `in` = Scanner(System.`in`)
    val delResources = `in`.nextLine()
    if (delResources.compareTo("y") == 0) {
        println("Lets clean up the AWS resources. Wait 30 seconds")
        TimeUnit.SECONDS.sleep(30)
        deleteEventSourceMapping(functionName)
        deleteSQSQueue(queueName)
        listBucketObjects(bucketName)
        deleteBucket(bucketName)
        delLambdaFunction(functionName)
        deleteLambdaRole(lambdaRoleName)
        deleteSagemakerRole(sageMakerRoleName)
        deletePipeline(pipelineName)
    } else {
        println("The AWS Resources were not deleted!")
    }
    println(DASHES)

    println(DASHES)
    println("SageMaker pipeline scenario is complete.")
    println(DASHES)
}

// Delete a SageMaker pipeline by name.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.deletePipeline(pipelineRequest)
        println("*** Successfully deleted $pipelineNameVal")
    }
```

```
}

suspend fun deleteSagemakerRole(roleNameVal: String) {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    IamClient { region = "us-west-2" }.use { iam ->
        for (policy in sageMakerRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteLambdaRole(roleNameVal: String) {
    val lambdaRolePolicies = getLambdaRolePolicies()
    IamClient { region = "us-west-2" }.use { iam ->
        for (policy in lambdaRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}
```

```
suspend fun delLambdaFunction(myFunctionName: String) {
    val request = DeleteFunctionRequest {
        functionName = myFunctionName
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request = DeleteBucketRequest {
        bucket = bucketName
    }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}

suspend fun deleteBucketObjects(bucketName: String, objectName: String?) {
    val toDelete = ArrayList<ObjectIdentifier>()
    val obId = ObjectIdentifier {
        key = objectName
    }
    toDelete.add(obId)
    val delOb = Delete {
        objects = toDelete
    }
    val dor = DeleteObjectsRequest {
        bucket = bucketName
        delete = delOb
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.deleteObjects(dor)
        println("*** $bucketName objects were deleted.")
    }
}

suspend fun listBucketObjects(bucketNameVal: String) {
    val listObjects = ListObjectsRequest {
        bucket = bucketNameVal
```

```
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val objects = res.contents
        if (objects != null) {
            for (myValue in objects) {
                println("The name of the key is ${myValue.key}")
                deleteBucketObjects(bucketNameVal, myValue.key)
            }
        }
    }
}

// Delete the specific Amazon SQS queue.
suspend fun deleteSQSQueue(queueNameVal: String?) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val urlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = urlVal
        }
        sqsClient.deleteQueue(deleteQueueRequest)
    }
}

// Delete the queue event mapping.
suspend fun deleteEventSourceMapping(functionNameVal: String) {
    if (eventSourceMapping.compareTo("") == 0) {
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
            val request = ListEventSourceMappingsRequest {
                functionName = functionNameVal
            }
            val response = lambdaClient.listEventSourceMappings(request)
            val eventList = response.eventSourceMappings
            if (eventList != null) {
                for (event in eventList) {
                    eventSourceMapping = event.uuid.toString()
                }
            }
        }
```

```
    }

    val eventSourceMappingRequest = DeleteEventSourceMappingRequest {
        uuid = eventSourceMapping
    }
    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest)
        println("The event mapping is deleted!")
    }
}

// Reads the objects in the S3 bucket and displays the values.
private suspend fun readObject(bucketName: String, keyVal: String?) {
    println("Output file contents: \n")
    val objectRequest = GetObjectRequest {
        bucket = bucketName
        key = keyVal
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.getObject(objectRequest) { resp ->
            val byteArray = resp.body?.toByteArray()
            val text = byteArray?.let { String(it, StandardCharsets.UTF_8) }
            println("Text output: $text")
        }
    }
}

// Display the results from the output directory.
suspend fun getOutputResults(bucketName: String?) {
    println("Getting output results $bucketName.")
    val listObjectsRequest = ListObjectsRequest {
        bucket = bucketName
        prefix = "outputfiles/"
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        val response = s3Client.listObjects(listObjectsRequest)
        val s3Objects: List<Object>? = response.contents
        if (s3Objects != null) {
            for (`object` in s3Objects) {
                if (bucketName != null) {
                    readObject(bucketName, (`object`.key))
                }
            }
        }
```

```
    }
}

suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
 sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}

// Start a pipeline run with job configurations.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn: String?,
 pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
        value = roleArn
    }
    val para2 = Parameter {
        name = "parameter_queue_url"
```

```
        value = queueUrl
    }

    val inputJSON = """{
        "DataSourceConfig": {
        "S3Data": {
            "S3Uri": "s3://$bucketName/samplefiles/latlongtest.csv"
        },
        "Type": "S3_DATA"
        },
        "DocumentType": "CSV"
    }"""
    println(inputJSON)
    val para3 = Parameter {
        name = "parameter_vej_input_config"
        value = inputJSON
    }

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    val jobS3Data = VectorEnrichmentJobS3Data {
        s3Uri = output
    }

    val outputConfig = ExportVectorEnrichmentJobOutputConfig {
        s3Data = jobS3Data
    }

    val gson4: String = gson.toJson(outputConfig)
    val para4: Parameter = Parameter {
        name = "parameter_vej_export_config"
        value = gson4
    }
    println("parameter_vej_export_config:" + gson.toJson(outputConfig))

    val para5JSON =
        "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName\":
\"Longitude\",\"YAttributeName\":\"Latitude\"}}"

    val para5: Parameter = Parameter {
        name = "parameter_step_1_vej_config"
        value = para5JSON
    }

    parameters.add(para1)
```

```
    parameters.add(para2)
    parameters.add(para3)
    parameters.add(para4)
    parameters.add(para5)

    val pipelineExecutionRequest = StartPipelineExecutionRequest {
        pipelineExecutionDescription = "Created using Kotlin SDK"
        pipelineExecutionDisplayName = "$pipelineName-example-execution"
        pipelineParameters = parameters
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
    }
}

// Create a pipeline from the example pipeline JSON.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
 String?, pipelineNameVal: String?) {
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
        println(jsonObject)

        // Create the pipeline.
        val pipelineRequest = CreatePipelineRequest {
            pipelineDescription = "Kotlin SDK example pipeline"
            roleArn = roleArnVal
            pipelineName = pipelineNameVal
            pipelineDefinition = jsonObject.toString()
```

```
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            sageMakerClient.createPipeline(pipelineRequest)
        }
    }
}

suspend fun putS3Object(bucketName: String, objectKey: String, objectPath: String) {
    val request = PutObjectRequest {
        bucket = bucketName
        key = objectKey
        body = File(objectPath).asByteStream()
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putObject(request)
        println("Successfully placed $objectKey into bucket $bucketName")
    }
}

suspend fun setupBucket(bucketName: String) {
    val request = CreateBucketRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun checkBucket(bucketName: String): Boolean {
    try {
        val headBucketRequest = HeadBucketRequest {
            bucket = bucketName
        }
        S3Client { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            println("$bucketName exists")
            return true
        }
    } catch (e: S3Exception) {
        println("Bucket does not exist")
```

```
    }
    return false
}

// Connect the queue to the Lambda function as an event source.
suspend fun connectLambda(queueUrlVal: String?, lambdaNameVal: String?) {
    println("Connecting the Lambda function and queue for the pipeline.")
    var queueArn = ""

    // Specify the attributes to retrieve.
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)
    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val queueAtts = response.attributes
        if (queueAtts != null) {
            for ((key, value) in queueAtts) {
                println("Key = $key, Value = $value")
                queueArn = value
            }
        }
    }
    val eventSourceMappingRequest = CreateEventSourceMappingRequest {
        eventSourceArn = queueArn
        functionName = lambdaNameVal
    }
    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        val response1 =
 lambdaClient.createEventSourceMapping(eventSourceMappingRequest)
        eventSourceMapping = response1.uuid.toString()
        println("The mapping between the event source and Lambda function was
 successful")
    }
}

// Set up the SQS queue to use with the pipeline.
suspend fun setupQueue(queueNameVal: String, lambdaNameVal: String): String {
    println("Setting up queue named $queueNameVal")
    val queueAtt: MutableMap<String, String> = HashMap()
```

```
    queueAtt.put("DelaySeconds", "5")
    queueAtt.put("ReceiveMessageWaitTimeSeconds", "5")
    queueAtt.put("VisibilityTimeout", "300")

    val createQueueRequest = CreateQueueRequest {
        queueName = queueNameVal
        attributes = queueAtt
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("\nGet queue url")
        val getQueueUrlResponse = sqsClient.getQueueUrl(GetQueueUrlRequest
 { queueName = queueNameVal })
        TimeUnit.SECONDS.sleep(15)
        connectLambda(getQueueUrlResponse.queueUrl, lambdaNameVal)
        println("Queue ready with Url " + getQueueUrlResponse.queueUrl)
        return getQueueUrlResponse.queueUrl.toString()
    }
}

// Checks to see if the Amazon SQS queue exists. If not, this method creates a new
 queue
// and returns the ARN value.
suspend fun checkQueue(queueNameVal: String, lambdaNameVal: String): String? {
    println("Checking to see if the queue exists. If not, a new queue will be
 created for use in this workflow.")
    var queueUrl: String
    try {
        val request = GetQueueUrlRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-west-2" }.use { sqsClient ->
            val response = sqsClient.getQueueUrl(request)
            queueUrl = response.queueUrl.toString()
            println(queueUrl)
        }
    } catch (e: SqsException) {
        println(e.message + " A new queue will be created")
        queueUrl = setupQueue(queueNameVal, lambdaNameVal)
    }
    return queueUrl
}
```

```
suspend fun createNewFunction(myFunctionName: String, s3BucketName: String, myS3Key:
 String, myHandler: String, myRole: String): String {
    val functionCode = FunctionCode {
        s3Bucket = s3BucketName
        s3Key = myS3Key
    }

    val request = CreateFunctionRequest {
        functionName = myFunctionName
        code = functionCode
        description = "Created by the Lambda Kotlin API"
        handler = myHandler
        role = myRole
        runtime = Runtime.Java11
        memorySize = 1024
        timeout = 200
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        println("${functionResponse.functionArn} was created")
        return functionResponse.functionArn.toString()
    }
}

suspend fun checkFunction(myFunctionName: String, s3BucketName: String, myS3Key:
 String, myHandler: String, myRole: String): String {
    println("Checking to see if the function exists. If not, a new AWS Lambda
 function will be created for use in this workflow.")
    var functionArn: String
    try {
        // Does this function already exist.
        val functionRequest = GetFunctionRequest {
            functionName = myFunctionName
        }
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
            val response = lambdaClient.getFunction(functionRequest)
            functionArn = response.configuration?.functionArn.toString()
            println("$functionArn exists")
        }
```

```
    } catch (e: LambdaException) {
        println(e.message + " A new function will be created")
        functionArn = createNewFunction(myFunctionName, s3BucketName, myS3Key,
 myHandler, myRole)
    }
    return functionArn
}

// Checks to see if the SageMaker role exists. If not, this method creates it.
suspend fun checkSageMakerRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be created
 for AWS SageMaker to use.")
    var roleArn: String
    try {
        val roleRequest = GetRoleRequest {
            roleName = roleNameVal
        }
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
    } catch (e: IamException) {
        println(e.message + " A new role will be created")
        roleArn = createSageMakerRole(roleNameVal)
    }
    return roleArn
}

suspend fun createSageMakerRole(roleNameVal: String): String {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    println("Creating a role to use with SageMaker.")
    val assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\"," +
        "\"sagemaker-geospatial.amazonaws.com\"," +
        "\"lambda.amazonaws.com\"," +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}," +
```

```
        "\"Action\": \"sts:AssumeRole\"" +
        "}]" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in sageMakerRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policy
            }
            iamClient.attachRolePolicy(attachRequest)
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15)
        System.out.println("Role ready with ARN ${roleResult.role?.arn}")
        return roleResult.role?.arn.toString()
    }
}

// Checks to see if the Lambda role exists. If not, this method creates it.
suspend fun checkLambdaRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be created
 for AWS Lambda to use.")
    var roleArn: String
    val roleRequest = GetRoleRequest {
        roleName = roleNameVal
    }

    try {
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
    } catch (e: IamException) {
```

```
        println(e.message + " A new role will be created")
        roleArn = createLambdaRole(roleNameVal)
    }

    return roleArn
}

private suspend fun createLambdaRole(roleNameVal: String): String {
    val lambdaRolePolicies = getLambdaRolePolicies()
    val assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\"," +
        "\"sagemaker-geospatial.amazonaws.com\"," +
        "\"lambda.amazonaws.com\"," +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in lambdaRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policy
            }
            iamClient.attachRolePolicy(attachRequest)
        }

        // Allow time for the role to be ready.
```

```
        TimeUnit.SECONDS.sleep(15)
        println("Role ready with ARN " + roleResult.role?.arn)
        return roleResult.role?.arn.toString()
    }
}

fun getLambdaRolePolicies(): Array<String?> {
    val lambdaRolePolicies = arrayOfNulls<String>(5)
    lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess"
    lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess"
    lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess"
    lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy"
    lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
 "AWSLambdaSQSQueueExecutionRole"
    return lambdaRolePolicies
}

fun getSageMakerRolePolicies(): Array<String?> {
    val sageMakerRolePolicies = arrayOfNulls<String>(3)
    sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess"
    sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess"
    sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess"
    return sageMakerRolePolicies
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

SDK for Kotlin을 사용한 Secrets Manager 예제

다음 코드 예제에서는 Secrets Manager와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`GetSecretValue`

다음 코드 예시에서는 `GetSecretValue`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getValue(secretName: String?) {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        val response = secretsClient.getSecretValue(valueRequest)
        val secret = response.secretString
        println("The secret value is $secret")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetSecretValue를 참조하십시오.

SDK for Kotlin을 사용한 Amazon SES 예제

다음 코드 예제에서는 Amazon SES와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시나리오

시나리오

DynamoDB 데이터를 추적하는 웹 애플리케이션 만들기

다음 코드 예제는 Amazon DynamoDB 테이블의 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 전송하는 웹 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon DynamoDB API를 사용하여 DynamoDB 작업 데이터를 추적하는 동적 웹 애플리케이션을 만드는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon SES

Amazon Redshift 데이터를 추적하는 웹 애플리케이션 만들기

다음 코드 예제는 Amazon Redshift 데이터베이스를 사용하여 작업 항목을 추적하고 보고하는 웹 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon Redshift 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

Amazon Redshift 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 Spring REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 GitHub에서 전체 예제를 참조하세요.

이 예시에서 사용되는 서비스

- Amazon Redshift
- Amazon SES

Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제는 Amazon Aurora Serverless 데이터베이스의 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 전송하는 웹 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon RDS 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

Amazon Aurora Serverless 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 Spring REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 GitHub에서 전체 예제를 참조하세요.

이 예시에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

이미지에서 객체 감지

다음 코드 예제는 Amazon Rekognition을 사용하여 이미지의 범주별로 객체를 감지하는 앱을 구축하는 방법을 보여줍니다.

SDK for Kotlin

Amazon Rekognition Kotlin API를 사용하여 Amazon Simple Storage Service (Amazon S3) 버킷에 있는 이미지에서 범주별로 객체를 식별하기 위해 Amazon Rekognition을 사용하여 앱을 만드는 방법을 보여줍니다. 이 앱은 Amazon Simple Email Service(Amazon SES)를 사용하여 결과와 함께 이메일 알림을 관리자에게 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하십시오.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES

SDK for Kotlin을 사용한 Amazon SNS 예제

다음 코드 예제에서는 Amazon SNS와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시작

Hello Amazon SNS

다음 코드 예제에서는 Amazon SNS 사용을 시작하는 방법을 보여줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListTopics를 참조하십시오.

주제

- 작업
- 시나리오

작업

`CreateTopic`

다음 코드 예시에서는 `CreateTopic`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateTopic를 참조하세요.

`DeleteTopic`

다음 코드 예시에서는 `DeleteTopic`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }
```

```
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteTopic를 참조하세요.

`GetTopicAttributes`

다음 코드 예시에서는 `GetTopicAttributes`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetTopicAttributes를 참조하세요.

`ListSubscriptions`

다음 코드 예시에서는 `ListSubscriptions`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listSNSSubscriptions() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListSubscriptions를 참조하세요.

`ListTopics`

다음 코드 예시에서는 `ListTopics`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listSNSTopics() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
```

```
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListTopics를 참조하세요.

`Publish`

다음 코드 예시에서는 `Publish`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun pubTopic(
    topicArnVal: String,
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Publish를 참조하세요.

`SetTopicAttributes`

다음 코드 예시에서는 `SetTopicAttributes`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 SetTopicAttributes를 참조하세요.

`Subscribe`

다음 코드 예시에서는 `Subscribe`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

이메일 주소로 주제 구독.

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

Lambda 함수에서 주제를 구독합니다.

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Subscribe를 참조하세요.

`TagResource`

다음 코드 예시에서는 `TagResource`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun addTopicTags(topicArn: String) {
    val tag =
        Tag {
            key = "Team"
            value = "Development"
        }

    val tag2 =
        Tag {
            key = "Environment"
            value = "Gamma"
        }

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)

    val request =
        TagResourceRequest {
            resourceArn = topicArn
            tags = tagList
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 TagResource를 참조하세요.

`Unsubscribe`

다음 코드 예시에서는 `Unsubscribe`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Unsubscribe를 참조하세요.

시나리오

DynamoDB 테이블에 데이터를 제출하기 위한 앱 구축

다음 코드 예제는 Amazon DynamoDB 테이블에 데이터를 제출하고 사용자가 테이블을 업데이트할 때 알리는 애플리케이션을 빌드하는 방법을 보여줍니다.

SDK for Kotlin

Amazon DynamoDB Kotlin API를 사용하여 데이터를 제출하고 Amazon SNS Kotlin API를 사용하여 문자 메시지를 보내는 기본 Android 애플리케이션을 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon SNS

Amazon SNS 애플리케이션 구축

다음 코드 예제는 구독 및 게시 기능이 있고 메시지를 번역하는 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon SNS Kotlin API를 사용하여 구독 및 게시 기능이 있는 애플리케이션을 생성하는 방법을 보여줍니다. 또한 이 예제 애플리케이션은 메시지를 번역합니다.

전체 소스 코드와 웹 앱을 생성하는 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

전체 소스 코드와 기본 Android 앱을 생성하는 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예시에서 사용되는 서비스

- Amazon SNS
- Amazon Translate

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예시에서 사용되는 서비스

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

SMS 문자 메시지 게시

다음 코드 예제에서는 Amazon SNS를 사용하여 SMS 메시지를 게시하는 방법을 보여줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Publish를 참조하세요.

대기열에 메시지 게시

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 주제(FIFO 또는 비 FIFO)를 생성합니다.
- 필터 적용 옵션을 사용하여 여러 개의 대기열로 주제를 구독합니다.
- 주제에 메시지를 게시합니다.
- 대기열에서 받은 메시지를 폴링합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
package com.example.sns

import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
```

```
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
```

```
    println("Welcome to the AWS SDK for Kotlin messaging with topics and queues.")
    println(
        """
                In this scenario, you will create an SNS topic and subscribe an SQS
 queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication and
 message filtering.
                Would you like to work with FIFO topics? (y/n)
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
    if (useFIFO.compareTo("y") == 0) {
        selectFIFO = true
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically generated
 from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
 delivered.
        For more information about deduplication, see https://docs.aws.amazon.com/
sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
```

```
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
            groupId = input.nextLine()
        }
    }
    println(DASHES)

    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended to
the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
```

```
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
         },
         "Action": "sqs:SendMessage",
                 "Resource": "$sqsQueueArn",
                 "Condition": {
             "ArnEquals": {
                 "aws:SourceArn": "$topicArn"
             }
         }
     }
     ]
     }"""
    setQueueAttr(sqsQueueUrl, policy)
    println(DASHES)

    println(DASHES)
    println("6. Subscribe to the SQS queue.")
    if (selectFIFO) {
        println(
            """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/latest/
dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
        )
        println("Would you like to filter messages for $sqsQueueName's subscription
 to the topic $topicName?  (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the following
 \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
```

```
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
                    else -> moreAns = true
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following \"tone
\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
 deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
```

```
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key to
 continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
```

```
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was deleted")
    }
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }
```

```
    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
```

```
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
 aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
```

```
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
```

```
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic " +
 topicArnVal + "\n" +
                     "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}
```

```
suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
```

```
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}
```

```
suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API reference의 다음 주제를 참조하세요.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

SDK for Kotlin을 사용한 Amazon SQS 예제

다음 코드 예제에서는 Amazon SQS와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시작

Hello Amazon SNS

다음 코드 예제에서는 Amazon SQS를 사용하여 시작하는 방법을 보여줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
package com.kotlin.sqs

import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.paginators.listQueuesPaginated
import kotlinx.coroutines.flow.transform

suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SqsClient { region = "us-east-1" }.use { sqsClient ->
```

```
        sqsClient
            .listQueuesPaginated { }
            .transform { it.queueUrls?.forEach { queue -> emit(queue) } }
            .collect { queue ->
                println("The Queue URL is $queue")
            }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListQueues를 참조하십시오.

주제

- 작업
- 시나리오

작업

`CreateQueue`

다음 코드 예시에서는 `CreateQueue`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createQueue(queueNameVal: String): String {
    println("Create Queue")
    val createQueueRequest =
        CreateQueueRequest {
            queueName = queueNameVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
```

```
        println("Get queue url")

        val getQueueUrlRequest =
            GetQueueUrlRequest {
                queueName = queueNameVal
            }

        val getQueueUrlResponse = sqsClient.getQueueUrl(getQueueUrlRequest)
        return getQueueUrlResponse.queueUrl.toString()
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateQueue를 참조하십시오.

`DeleteMessage`

다음 코드 예시에서는 `DeleteMessage`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}
```

```
suspend fun deleteQueue(queueUrlVal: String) {
    val request =
        DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteMessage를 참조하십시오.

`DeleteQueue`

다음 코드 예시에서는 `DeleteQueue`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}
```

```
suspend fun deleteQueue(queueUrlVal: String) {
    val request =
        DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteQueue를 참조하십시오.

`ListQueues`

다음 코드 예시에서는 `ListQueues`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listQueues() {
    println("\nList Queues")

    val prefix = "que"
    val listQueuesRequest =
        ListQueuesRequest {
            queueNamePrefix = prefix
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.listQueues(listQueuesRequest)
        response.queueUrls?.forEach { url ->
            println(url)
        }
    }
```

```
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListQueues를 참조하십시오.

`ReceiveMessage`

다음 코드 예시에서는 `ReceiveMessage`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun receiveMessages(queueUrlVal: String?) {
    println("Retrieving messages from $queueUrlVal")

    val receiveMessageRequest =
        ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.receiveMessage(receiveMessageRequest)
        response.messages?.forEach { message ->
            println(message.body)
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ReceiveMessage를 참조하십시오.

`SendMessage`

다음 코드 예시에서는 `SendMessage`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun sendMessages(
    queueUrlVal: String,
    message: String,
) {
    println("Sending multiple messages")
    println("\nSend message")
    val sendRequest =
        SendMessageRequest {
            queueUrl = queueUrlVal
            messageBody = message
            delaySeconds = 10
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessage(sendRequest)
        println("A single message was successfully sent.")
    }
}

suspend fun sendBatchMessages(queueUrlVal: String?) {
    println("Sending multiple messages")

    val msg1 =
        SendMessageBatchRequestEntry {
            id = "id1"
            messageBody = "Hello from msg 1"
        }

    val msg2 =
        SendMessageBatchRequestEntry {
            id = "id2"
            messageBody = "Hello from msg 2"
        }
```

```
    val sendMessageBatchRequest =
        SendMessageBatchRequest {
            queueUrl = queueUrlVal
            entries = listOf(msg1, msg2)
        }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessageBatch(sendMessageBatchRequest)
        println("Batch message were successfully sent.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 SendMessage를 참조하십시오.

시나리오

메시징 애플리케이션 생성

다음 코드 예제에서는 Amazon SQS를 사용하여 메시징 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon SQS API를 사용하여 메시지를 보내고 검색하는 Spring REST API를 개발하는 방법을 보여 줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하십시오.

이 예시에서 사용되는 서비스

- Amazon Comprehend
- Amazon SQS

대기열에 메시지 게시

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 주제(FIFO 또는 비 FIFO)를 생성합니다.
- 필터 적용 옵션을 사용하여 여러 개의 대기열로 주제를 구독합니다.
- 주제에 메시지를 게시합니다.
- 대기열에서 받은 메시지를 폴링합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
package com.example.sns

import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:
```

```
 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and queues.")
    println(
        """
                In this scenario, you will create an SNS topic and subscribe an SQS
 queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
        """.trimIndent(),
    )
    println(DASHES)
```

```
    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication and
 message filtering.
                Would you like to work with FIFO topics? (y/n)
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
    if (useFIFO.compareTo("y") == 0) {
        selectFIFO = true
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically generated
 from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
 delivered.
        For more information about deduplication, see https://docs.aws.amazon.com/
sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
            groupId = input.nextLine()
        }
    }
    println(DASHES)

    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
```

```
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended to
the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
     "Statement": [
     {
         "Effect": "Allow",
                "Principal": {
             "Service": "sns.amazonaws.com"
         },
         "Action": "sqs:SendMessage",
                "Resource": "$sqsQueueArn",
```

```
                "Condition": {
             "ArnEquals": {
                 "aws:SourceArn": "$topicArn"
             }
         }
     }
     ]
     }"""
    setQueueAttr(sqsQueueUrl, policy)
    println(DASHES)

    println(DASHES)
    println("6. Subscribe to the SQS queue.")
    if (selectFIFO) {
        println(
            """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/latest/
dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
        )
        println("Would you like to filter messages for $sqsQueueName's subscription
 to the topic $topicName?  (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the following
 \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
                    else -> moreAns = true
                }
            }
        }
```

```
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following \"tone
\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
 deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
```

```
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key to
 continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was deleted")
    }
}

suspend fun deleteSQSQueue(queueNameVal: String) {
```

```
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}
```

```
suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
```

```
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
```

```
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic " +
 topicArnVal + "\n" +
                    "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
```

```
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
```

```
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }
```

```
            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
```

```
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API reference의 다음 주제를 참조하세요.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

SDK for Kotlin을 사용한 Step Functions의 예제

다음 코드 예제에서는 Step Functions와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시작

Hello Step Functions

다음 코드 예제에서는 Step Functions를 사용하여 시작하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest

/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

suspend fun main() {
    println(DASHES)
    println("Welcome to the AWS Step Functions Hello example.")
    println("Lets list up to ten of your state machines:")
    println(DASHES)

    listMachines()
}

suspend fun listMachines() {
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListStateMachines를 참조하십시오.

주제

- 기본 사항
- 작업

기본 사항

기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 활동을 생성합니다.
- 이전에 생성한 활동을 한 단계로 포함하는 Amazon States Language 정의에서 상태 시스템을 생성합니다.
- 상태 시스템을 실행하고 사용자 입력으로 활동에 응답합니다.
- 실행 완료 후 최종 상태 및 출력을 가져온 다음 리소스를 정리합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import aws.sdk.kotlin.services.iam.IamClient
import aws.sdk.kotlin.services.iam.model.CreateRoleRequest
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.CreateActivityRequest
import aws.sdk.kotlin.services.sfn.model.CreateStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DeleteActivityRequest
import aws.sdk.kotlin.services.sfn.model.DeleteStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DescribeExecutionRequest
import aws.sdk.kotlin.services.sfn.model.DescribeStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.GetActivityTaskRequest
import aws.sdk.kotlin.services.sfn.model.ListActivitiesRequest
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest
```

```
import aws.sdk.kotlin.services.sfn.model.SendTaskSuccessRequest
import aws.sdk.kotlin.services.sfn.model.StartExecutionRequest
import aws.sdk.kotlin.services.sfn.model.StateMachineType
import aws.sdk.kotlin.services.sfn.paginators.listActivitiesPaginated
import aws.sdk.kotlin.services.sfn.paginators.listStateMachinesPaginated
import com.fasterxml.jackson.databind.JsonNode
import com.fasterxml.jackson.databind.ObjectMapper
import com.fasterxml.jackson.databind.node.ObjectNode
import kotlinx.coroutines.flow.transform
import java.util.Scanner
import java.util.UUID
import kotlin.collections.ArrayList
import kotlin.system.exitProcess

/**
 To run this code example, place the chat_sfn_state_machine.json file into your
 project's resources folder.

 You can obtain the JSON file to create a state machine in the following GitHub
 location:

 https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files

 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin code example performs the following tasks:

 1. List activities using a paginator.
 2. List state machines using a paginator.
 3. Creates an activity.
 4. Creates a state machine.
 5. Describes the state machine.
 6. Starts execution of the state machine and interacts with it.
 7. Describes the execution.
 8. Deletes the activity.
 9. Deletes the state machine.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
```

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <roleARN> <activityName> <stateMachineName>

    Where:
        roleName - The name of the IAM role to create for this state machine.
        activityName - The name of an activity to create.
        stateMachineName - The name of the state machine to create.
        jsonFile - The location of the chat_sfn_state_machine.json file. You can
 located it in resources/sample_files.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(0)
    }

    val roleName = args[0]
    val activityName = args[1]
    val stateMachineName = args[2]
    val jsonFile = args[3]
    val sc = Scanner(System.`in`)
    var action = false

    val polJSON = """{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "states.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
    }"""

    println(DASHES)
    println("Welcome to the AWS Step Functions example scenario.")
    println(DASHES)

    println(DASHES)
```

```
    println("1. List activities using a Paginator.")
    listActivitesPagnator()
    println(DASHES)

    println(DASHES)
    println("2. List state machines using a paginator.")
    listStatemachinesPagnator()
    println(DASHES)

    println(DASHES)
    println("3. Create a new activity.")
    val activityArn = createActivity(activityName)
    println("The ARN of the Activity is $activityArn")
    println(DASHES)

    // Get JSON to use for the state machine and place the activityArn value into
it.
    val stream = GetStream()
    val jsonString = stream.getStream(jsonFile)

    // Modify the Resource node.
    val objectMapper = ObjectMapper()
    val root: JsonNode = objectMapper.readTree(jsonString)
    (root.path("States").path("GetInput") as ObjectNode).put("Resource",
activityArn)

    // Convert the modified Java object back to a JSON string.
    val stateDefinition = objectMapper.writeValueAsString(root)
    println(stateDefinition)

    println(DASHES)
    println("4. Create a state machine.")
    val roleARN = createIAMRole(roleName, polJSON)
    val stateMachineArn = createMachine(roleARN, stateMachineName, stateDefinition)
    println("The ARN of the state machine is $stateMachineArn")
    println(DASHES)

    println(DASHES)
    println("5. Describe the state machine.")
    describeStateMachine(stateMachineArn)
    println("What should ChatSFN call you?")
    val userName = sc.nextLine()
    println("Hello $userName")
    println(DASHES)
```

```
    println(DASHES)
    // The JSON to pass to the StartExecution call.
    val executionJson = "{ \"name\" : \"$userName\" }"
    println(executionJson)
    println("6. Start execution of the state machine and interact with it.")
    val runArn = startWorkflow(stateMachineArn, executionJson)
    println("The ARN of the state machine execution is $runArn")
    var myList: List<String>
    while (!action) {
        myList = getActivityTask(activityArn)
        println("ChatSFN: " + myList[1])
        println("$userName please specify a value.")
        val myAction = sc.nextLine()
        if (myAction.compareTo("done") == 0) {
            action = true
        }
        println("You have selected $myAction")
        val taskJson = "{ \"action\" : \"$myAction\" }"
        println(taskJson)
        sendTaskSuccess(myList[0], taskJson)
    }
    println(DASHES)

    println(DASHES)
    println("7. Describe the execution.")
    describeExe(runArn)
    println(DASHES)

    println(DASHES)
    println("8. Delete the activity.")
    deleteActivity(activityArn)
    println(DASHES)

    println(DASHES)
    println("9. Delete the state machines.")
    deleteMachine(stateMachineArn)
    println(DASHES)

    println(DASHES)
    println("The AWS Step Functions example scenario is complete.")
    println(DASHES)
}
```

```
suspend fun listStatemachinesPagnator() {
    val machineRequest =
        ListStateMachinesRequest {
            maxResults = 10
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient
            .listStateMachinesPaginated(machineRequest)
            .transform { it.stateMachines?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" The state machine ARN is ${obj.stateMachineArn}")
            }
    }
}

suspend fun listActivitesPagnator() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient
            .listActivitiesPaginated(activitiesRequest)
            .transform { it.activities?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" The activity ARN is ${obj.activityArn}")
            }
    }
}

suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}
```

```
suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}

suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient { region = "us-east-1" }.use { sfnClient ->
            val response = sfnClient.describeExecution(executionRequest)
            status = response.status.toString()
            if (status.compareTo("Running") == 0) {
                println("The state machine is still running, let's wait for it to
 finish.")
                Thread.sleep(2000)
            } else if (status.compareTo("Succeeded") == 0) {
                println("The Step Function workflow has succeeded")
                hasSucceeded = true
            } else {
                println("The Status is $status")
            }
        }
    }
    println("The Status is $status")
}

suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
```

```
        SendTaskSuccessRequest {
            taskToken = token
            output = json
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}

suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
}

suspend fun startWorkflow(
    stateMachineArnVal: String?,
    jsonEx: String?,
): String? {
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val executionRequest =
        StartExecutionRequest {
            input = jsonEx
            stateMachineArn = stateMachineArnVal
            name = uuidValue
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}

suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
```

```
            stateMachineArn = stateMachineArnVal
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.describeStateMachine(stateMachineRequest)
        println("The name of the State machine is ${response.name}")
        println("The status of the State machine is ${response.status}")
        println("The ARN value of the State machine is ${response.stateMachineArn}")
        println("The role ARN value is ${response.roleArn}")
    }
}

suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}

suspend fun createIAMRole(
    roleNameVal: String?,
    polJSON: String?,
): String? {
    val request =
        CreateRoleRequest {
            roleName = roleNameVal
            assumeRolePolicyDocument = polJSON
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
```

```
    }
}

suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines
  - SendTaskSuccess
  - StartExecution
  - StopExecution

작업

`CreateActivity`

다음 코드 예시에서는 `CreateActivity`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateActivity를 참조하십시오.

`CreateStateMachine`

다음 코드 예시에서는 `CreateStateMachine`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
```

```
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateStateMachine을 참조하십시오.

`DeleteActivity`

다음 코드 예시에서는 `DeleteActivity`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteActivity를 참조하십시오.

`DeleteStateMachine`

다음 코드 예시에서는 `DeleteStateMachine`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteStateMachine을 참조하십시오.

`DescribeExecution`

다음 코드 예시에서는 `DescribeExecution`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient { region = "us-east-1" }.use { sfnClient ->
            val response = sfnClient.describeExecution(executionRequest)
            status = response.status.toString()
            if (status.compareTo("Running") == 0) {
                println("The state machine is still running, let's wait for it to
 finish.")
                Thread.sleep(2000)
            } else if (status.compareTo("Succeeded") == 0) {
                println("The Step Function workflow has succeeded")
                hasSucceeded = true
            } else {
                println("The Status is $status")
            }
        }
    }
    println("The Status is $status")
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeExecution을 참조하십시오.

`DescribeStateMachine`

다음 코드 예시에서는 `DescribeStateMachine`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.describeStateMachine(stateMachineRequest)
        println("The name of the State machine is ${response.name}")
        println("The status of the State machine is ${response.status}")
        println("The ARN value of the State machine is ${response.stateMachineArn}")
        println("The role ARN value is ${response.roleArn}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeStateMachine를 참조하십시오.

`GetActivityTask`

다음 코드 예시에서는 `GetActivityTask`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
```

```
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetActivityTask를 참조하십시오.

`ListActivities`

다음 코드 예시에서는 `ListActivities`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listAllActivites() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listActivities(activitiesRequest)
        response.activities?.forEach { item ->
            println("The activity ARN is ${item.activityArn}")
            println("The activity name is ${item.name}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListActivities를 참조하십시오.

`ListExecutions`

다음 코드 예시에서는 `ListExecutions`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getExeHistory(exeARN: String?) {
    val historyRequest =
        GetExecutionHistoryRequest {
            executionArn = exeARN
            maxResults = 10
        }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getExecutionHistory(historyRequest)
        response.events?.forEach { event ->
            println("The event type is ${event.type}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListExecutions를 참조하십시오.

`ListStateMachines`

다음 코드 예시에서는 `ListStateMachines`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import aws.sdk.kotlin.services.sfn.SfnClient
```

```
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest

/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

suspend fun main() {
    println(DASHES)
    println("Welcome to the AWS Step Functions Hello example.")
    println("Lets list up to ten of your state machines:")
    println(DASHES)

    listMachines()
}

suspend fun listMachines() {
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListStateMachines를 참조하십시오.

`SendTaskSuccess`

다음 코드 예시에서는 `SendTaskSuccess`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
            output = json
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 SendTaskSuccess를 참조하십시오.

`StartExecution`

다음 코드 예시에서는 `StartExecution`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun startWorkflow(
    stateMachineArnVal: String?,
    jsonEx: String?,
): String? {
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val executionRequest =
        StartExecutionRequest {
            input = jsonEx
            stateMachineArn = stateMachineArnVal
            name = uuidValue
        }
```

```
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 StartExecution를 참조하십시오.

지원 SDK for Kotlin을 사용한 예제

다음 코드 예제에서는 Kotlin용 AWS SDK를와 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 지원.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시작

안녕하세요 지원

다음 코드 예제에서는 지원의 사용을 시작하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.
```

```
For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

In addition, you must have the AWS Business Support Plan to use the AWS Support Java
 API. For more information, see:

https://aws.amazon.com/premiumsupport/plans/

This Kotlin example performs the following task:

1. Gets and displays available services.
 */

suspend fun main() {
    displaySomeServices()
}

// Return a List that contains a Service name and Category name.
suspend fun displaySomeServices() {
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is: " + service.name)

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                index++
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeServices를 참조하십시오.

주제

- 기본 사항
- 작업

기본 사항

기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용 가능한 서비스 및 사례의 심각도 수준을 가져와서 표시합니다.
- 선택한 서비스, 범주 및 심각도 수준을 사용하여 지원 사례를 만듭니다.
- 현재 일자의 미해결 사례 목록을 가져와서 표시합니다.
- 새로운 사례에 첨부 파일 세트와 통신을 추가합니다.
- 해당 사례에 대한 새로운 첨부 파일과 통신을 설명하세요.
- 사건을 해결하세요.
- 현재 일자의 해결된 사례 목록을 가져와서 표시합니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.
```

```
For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
In addition, you must have the AWS Business Support Plan to use the AWS Support Java
 API. For more information, see:

https://aws.amazon.com/premiumsupport/plans/

This Kotlin example performs the following tasks:
1. Gets and displays available services.
2. Gets and displays severity levels.
3. Creates a support case by using the selected service, category, and severity
 level.
4. Gets a list of open cases for the current day.
5. Creates an attachment set with a generated file.
6. Adds a communication with the attachment to the support case.
7. Lists the communications of the support case.
8. Describes the attachment set included with the communication.
9. Resolves the support case.
10. Gets a list of resolved cases for the current day.
*/

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <fileAttachment>
    Where:
         fileAttachment - The file can be a simple saved .txt file to use as an
 email attachment.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val fileAttachment = args[0]
    println("***** Welcome to the AWS Support case example scenario.")
    println("***** Step 1. Get and display available services.")
    val sevCatList = displayServices()

    println("***** Step 2. Get and display Support severity levels.")
    val sevLevel = displaySevLevels()
```

```
    println("***** Step 3. Create a support case using the selected service,
 category, and severity level.")
    val caseIdVal = createSupportCase(sevCatList, sevLevel)
    if (caseIdVal != null) {
        println("Support case $caseIdVal was successfully created!")
    } else {
        println("A support case was not successfully created!")
        exitProcess(1)
    }

    println("***** Step 4. Get open support cases.")
    getOpenCase()

    println("***** Step 5. Create an attachment set with a generated file to add to
 the case.")
    val attachmentSetId = addAttachment(fileAttachment)
    println("The Attachment Set id value is $attachmentSetId")

    println("***** Step 6. Add communication with the attachment to the support
 case.")
    addAttachSupportCase(caseIdVal, attachmentSetId)

    println("***** Step 7. List the communications of the support case.")
    val attachId = listCommunications(caseIdVal)
    println("The Attachment id value is $attachId")

    println("***** Step 8. Describe the attachment set included with the
 communication.")
    describeAttachment(attachId)

    println("***** Step 9. Resolve the support case.")
    resolveSupportCase(caseIdVal)

    println("***** Step 10. Get a list of resolved cases for the current day.")
    getResolvedCase()
    println("***** This Scenario has successfully completed")
}

suspend fun getResolvedCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
```

```
        DescribeCasesRequest {
            maxResults = 30
            afterTime = yesterday.toString()
            beforeTime = now.toString()
            includeResolvedCases = true
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}

suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}

suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
            caseId = caseIdVal
```

```
            maxResults = 10
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}

suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS Support
 case")
        } else {
            println("There was an error adding the communication to an AWS Support
 case")
        }
    }
}

suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal =
        Attachment {
            fileName = myFile.name
```

```
            data = sourceBytes
        }

    val setRequest =
        AddAttachmentsToSetRequest {
            attachments = listOf(attachmentVal)
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}

suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 20
            afterTime = yesterday.toString()
            beforeTime = now.toString()
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun createSupportCase(
    sevCatListVal: List<String>,
    sevLevelVal: String,
): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest =
        CreateCaseRequest {
```

```
            categoryCode = caseCategory.lowercase(Locale.getDefault())
            serviceCode = serCode.lowercase(Locale.getDefault())
            severityCode = sevLevelVal.lowercase(Locale.getDefault())
            communicationBody = "Test issue with
 ${serCode.lowercase(Locale.getDefault())}"
            subject = "Test case, please ignore"
            language = "en"
            issueType = "technical"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}

suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
        return levelName
    }
}

// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }
```

```
    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications

- DescribeServices
- DescribeSeverityLevels
- ResolveCase

작업

`AddAttachmentsToSet`

다음 코드 예시에서는 AddAttachmentsToSet을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal =
        Attachment {
            fileName = myFile.name
            data = sourceBytes
        }

    val setRequest =
        AddAttachmentsToSetRequest {
            attachments = listOf(attachmentVal)
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 AddAttachmentsToSet를 참조하십시오.

`AddCommunicationToCase`

다음 코드 예시에서는 `AddCommunicationToCase`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS Support
 case")
        } else {
            println("There was an error adding the communication to an AWS Support
 case")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 AddCommunicationToCase를 참조하십시오.

`CreateCase`

다음 코드 예시에서는 `CreateCase`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createSupportCase(
    sevCatListVal: List<String>,
    sevLevelVal: String,
): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest =
        CreateCaseRequest {
            categoryCode = caseCategory.lowercase(Locale.getDefault())
            serviceCode = serCode.lowercase(Locale.getDefault())
            severityCode = sevLevelVal.lowercase(Locale.getDefault())
            communicationBody = "Test issue with
 ${serCode.lowercase(Locale.getDefault())}"
            subject = "Test case, please ignore"
            language = "en"
            issueType = "technical"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}
```

- API에 세부 정보는 AWS SDK for Kotlin API 참조의 CreateCase를 참조하십시오.

`DescribeAttachment`

다음 코드 예시에서는 `DescribeAttachment`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeAttachment를 참조하십시오.

`DescribeCases`

다음 코드 예시에서는 `DescribeCases`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
```

```
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 20
            afterTime = yesterday.toString()
            beforeTime = now.toString()
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeCases를 참조하십시오.

`DescribeCommunications`

다음 코드 예시에서는 `DescribeCommunications`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
            caseId = caseIdVal
            maxResults = 10
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCommunications(communicationsRequest)
```

```
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeCommunications를 참조하십시오.

`DescribeServices`

다음 코드 예시에서는 `DescribeServices`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
```

```
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeServices를 참조하십시오.

`DescribeSeverityLevels`

다음 코드 예시에서는 `DescribeSeverityLevels`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun displaySevLevels(): String {
    var levelName = ""
```

```
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
        return levelName
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeSeverityLevels를 참조하십시오.

`ResolveCase`

다음 코드 예시에서는 `ResolveCase`을 사용하는 방법을 보여 줍니다.

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ResolveCase를 참조하십시오.

SDK for Kotlin을 사용한 Amazon Translate 예제

다음 코드 예제에서는 Amazon Translate와 함께 AWS SDK for Kotlin을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시나리오

시나리오

Amazon SNS 애플리케이션 구축

다음 코드 예제는 구독 및 게시 기능이 있고 메시지를 번역하는 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Kotlin

Amazon SNS Kotlin API를 사용하여 구독 및 게시 기능이 있는 애플리케이션을 생성하는 방법을 보여줍니다. 또한 이 예제 애플리케이션은 메시지를 번역합니다.

전체 소스 코드와 웹 앱을 생성하는 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

전체 소스 코드와 기본 Android 앱을 생성하는 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예시에서 사용되는 서비스

- Amazon SNS
- Amazon Translate

에 대한 보안 AWS SDK for Kotlin

Amazon Web Services(AWS)에서 가장 우선순위가 높은 것이 클라우드 보안입니다. AWS 고객으로서 여러분은 가장 높은 보안 요구 사항을 충족하기 위해 설계된 데이터 센터 및 네트워크 아키텍처의 혜택을 받게 됩니다. 보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

클라우드 보안 - AWS 는 클라우드에서 제공되는 모든 서비스를 실행하는 인프라를 보호하고 안전하게 사용할 수 있는 서비스를 AWS 제공할 책임이 있습니다. 보안 책임은에서 가장 중요하며 AWS, 보안의 효율성은 AWS 규정 준수 프로그램의 일환으로 타사 감사자가 정기적으로 테스트하고 확인합니다.

클라우드의 보안 - 사용자의 책임은 사용 중인 AWS 서비스와 데이터의 민감도, 조직의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요인에 따라 결정됩니다.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 AWS 서비스 보안 설명서 페이지 및 규정 AWSAWS 준수 프로그램의 규정 준수 노력 범위에 속하는 서비스를 참조하세요.

주제

- 의 데이터 보호 AWS SDK for Kotlin
- AWS SDK for Kotlin TLS 1.2 지원
- ID 및 액세스 관리
- 이 AWS 제품 또는 서비스에 대한 규정 준수 검증
- 이 AWS 제품 또는 서비스에 대한 복원력
- 이 AWS 제품 또는 서비스에 대한 인프라 보안

의 데이터 보호 AWS SDK for Kotlin

AWS 공동 책임 모델의 데이터 보호에 적용됩니다 AWS SDK for Kotlin. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- AWS 암호화 솔루션과 내부의 모든 기본 보안 제어를 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 SDK for Kotlin 또는 기타 AWS 서비스 에서 콘솔, API AWS CLI또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명을 URL에 포함해서는 안 됩니다.

AWS SDK for Kotlin TLS 1.2 지원

다음 정보는 Java SSL 구현(JVM을 AWS SDK for Kotlin 대상으로 하는의 기본 SSL 구현)에만 적용됩니다. 다른 SSL 구현을 사용하는 경우 해당 SSL 구현을 참조하여 TLS 버전을 적용하는 방법을 알아보십시오.

Java에서의 TLS 지원

TLS 1.2는 Java 7부터 지원됩니다.

TLS 버전을 확인하는 방법

Java 가상 머신(JVM)에서 지원되는 TLS 버전을 확인하려면 다음 코드를 사용할 수 있습니다.

```
println(SSLContext.getDefault().supportedSSLParameters.protocols.joinToString(separator
 = ", "))
```

작동 중인 SSL 핸드셰이크와 사용된 TLS 버전을 보려면 시스템 속성 javax.net.debug를 사용하면 됩니다.

```
-Djavax.net.debug=ssl
```

ID 및 액세스 관리

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 도와주는 입니다. IAM 관리자는 AWS 리소스를 사용할 수 있는 인증(로그인) 및 권한 부여(권한 있음)를 받을 수 있는 사용자를 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

- 대상
- ID를 통한 인증
- 정책을 사용하여 액세스 관리
- IAM AWS 서비스 작업 방식
- AWS 자격 증명 및 액세스 문제 해결

대상

사용 방법 AWS Identity and Access Management (IAM)은에서 수행하는 작업에 따라 다릅니다 AWS.

서비스 사용자 - AWS 서비스 를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관리자가 제공합니다. 더 많은 AWS 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방법을 이해하면 관리자에게 올바른 권한을 요청하는 데 도움이 됩니다. 에서 기능에 액세스할 수 없는 경우 사용 중인의 AWS 자격 증명 및 액세스 문제 해결 또는 사용 설명서를 AWS참조 AWS 서비스 하세요.

서비스 관리자 - 회사에서 AWS 리소스를 책임지고 있는 경우에 대한 전체 액세스 권한이 있을 수 있습니다 AWS. 서비스 사용자가 액세스해야 하는 AWS 기능과 리소스를 결정하는 것은 사용자의 작업입니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지

의 정보를 검토하여 IAM의 기본 개념을 이해하세요. 회사가 IAM을와 함께 사용하는 방법에 대한 자세한 내용은 사용 중인의 AWS 서비스 사용 설명서를 AWS참조하세요.

IAM 관리자 - IAM 관리자라면 AWS에 대한 액세스 권한 관리 정책 작성 방법을 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 자격 AWS 증명 기반 정책 예제를 보려면 사용 중인의 사용 설명서를 참조 AWS 서비스 하세요.

ID를 통한 인증

인증은 자격 증명 AWS 으로에 로그인하는 방법입니다. IAM 사용자 또는 AWS 계정 루트 사용자 IAM 역할을 수임하여 로 인증(로그인 AWS)되어야 합니다.

자격 증명 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 자격 증명 AWS 으로에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증 및 Google 또는 Facebook 자격 증명은 페더레이션 자격 증명의 예입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 AWS 에 액세스하면 간접적으로 역할을 수임하게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의에 로그인하는 방법을 AWS참조하세요. AWS 계정

AWS 프로그래밍 방식으로에 액세스하는 경우는 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK)와 명령줄 인터페이스(CLI)를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 직접 요청에 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 자세한 방법은 IAM 사용 설명서에서 API 요청용AWS Signature Version 4를 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어는 다중 인증(MFA)을 사용하여 계정의 보안을 강화할 것을 AWS 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 다중 인증 및 IAM 사용 설명서에서 IAM의AWS 다중 인증을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 이 자격 증명을 AWS 계정 테루트 사용자라고 하며 계정을 생성하는 데 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명을 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 보안 인증이 필요한 작업을 참조하세요.

페더레이션 자격 증명

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 포함한 인간 사용자가 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 AWS 서비스 에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 사용자 디렉터리, 웹 자격 증명 공급자, AWS Directory Service, Identity Center 디렉터리 또는 자격 증명 소스를 통해 제공된 자격 증명을 사용하여 AWS 서비스 에 액세스하는 모든 사용자의 사용자입니다. 페더레이션 자격 증명에 액세스할 때 역할을 AWS 계정수임하고 역할은 임시 자격 증명을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center을(를) 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 모든 및 애플리케이션에서 사용할 수 있도록 자체 자격 증명 소스의 사용자 AWS 계정 및 그룹 집합에 연결하고 동기화할 수 있습니다. IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇인가요?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 한 사람 또는 애플리케이션에 대한 특정 권한이 AWS 계정 있는 내 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우, 정기적으로 액세스 키 교체를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 자격 증명을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 내용은 IAM 사용 설명서에서 IAM 사용자 사용 사례를 참조하세요.

IAM 역할

IAM 역할은 특정 권한이 AWS 계정 있는 내 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 에서 IAM 역할을 일시적으로 수임하려면 사용자에서 IAM 역할(콘솔)로 전환할 AWS Management Console수 있습니다. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-console.html 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 AWS

CLI 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 관련 역할에 대한 자세한 내용은 IAM 사용 설명서의 Create a role for a third-party identity provider (federation)를 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할과 연관짓습니다. 권한 집합에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 집합을 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 교차 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 에서는 (역할을 프록시로 사용하는 대신) 정책을 리소스에 직접 연결할 AWS 서비스 수 있습니다. 교차 계정 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하세요.
- 교차 서비스 액세스 - 일부는 다른에서 기능을 AWS 서비스 사용합니다 AWS 서비스. 예를 들어, 서비스에서 호출하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 위탁자의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 다른 AWS 서비스 또는 리소스와의 상호 작용을 완료해야 하는 요청을 수신할 때만 수행됩니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 Create a role to delegate permissions to an AWS 서비스를 참조하세요.
  - 서비스 연결 역할 - 서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

- Amazon EC2에서 실행되는 애플리케이션 - IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로필에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 AWS 때 권한을 정의하는의 객체입니다.는 보안 주체(사용자, 루트 사용자 또는 역할 세션)가 요청할 때 이러한 정책을 AWS 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 대상에 액세스할 수 있는 사용자를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자 및 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console AWS CLI, 또는 API에서 역할 정보를 가져올 수 있습니다 AWS .

ID 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은의 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립 실행형 정책입니다 AWS 계정. 관리형 정책에는 AWS 관리형 정책 및 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 위탁자가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 위탁자를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는가 포함될 수 있습니다 AWS 서비스.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3 AWS WAF및 Amazon VPC는 ACLs. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

AWS 는 덜 일반적인 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 ID 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 객체의 ID 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔티티에 대한 권한 경계를 참조하세요.
- 서비스 제어 정책(SCPs) - SCPs는의 조직 또는 조직 단위(OU)에 대한 최대 권한을 지정하는 JSON 정책입니다 AWS Organizations. AWS Organizations 는 비즈니스가 소유 AWS 계정 한 여러를 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 기능을 활성화할 경우, 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각각을 포함하여 멤버 계정의 엔터티에 대한 권한을 제한합니다 AWS 계정 루트 사용자. 조직 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서에서 Service control policies을 참조하세요.

- 리소스 제어 정책(RCP) - RCP는 소유한 각 리소스에 연결된 IAM 정책을 업데이트하지 않고 계정의 리소스에 대해 사용 가능한 최대 권한을 설정하는 데 사용할 수 있는 JSON 정책입니다. RCP는 멤버 계정의 리소스에 대한 권한을 제한하며 조직에 속하는지 여부에 AWS 계정 루트 사용자관계없이를 포함한 자격 증명에 대한 유효 권한에 영향을 미칠 수 있습니다. RCP를 AWS 서비스 지원하는 목록을 포함하여 조직 및 RCPs에 대한 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCPs)을 참조하세요.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 ID 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

IAM AWS 서비스 작업 방식

대부분의 IAM 기능을 AWS 서비스 사용하는 방법을 개괄적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

특정를 IAM과 AWS 서비스 함께 사용하는 방법을 알아보려면 관련 서비스 사용 설명서의 보안 섹션을 참조하세요.

AWS 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 및 IAM으로 작업할 때 발생할 수 있는 일반적인 문제를 진단 AWS 하고 수정할 수 있습니다.

주제

- 에서 작업을 수행할 권한이 없음 AWS
- iam:PassRole을 수행하도록 인증되지 않음
- 내 외부의 사람이 내 AWS 리소스에 액세스 AWS 계정 하도록 허용하고 싶습니다.

에서 작업을 수행할 권한이 없음 AWS

작업을 수행할 권한이 없다는 오류가 수신되면, 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음의 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 awes:*GetWidget* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

이 경우, awes:*GetWidget* 작업을 사용하여 *my-example-widget* 리소스에 액세스할 수 있도록 mateojackson 사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

iam:PassRole을 수행하도록 인증되지 않음

iam:PassRole 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 AWS에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예 오류는 marymajor라는 IAM 사용자가 콘솔을 사용하여 AWS에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 iam:PassRole 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 외부의 사람이 내 AWS 리소스에 액세스 AWS 계정 하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- 에서 이러한 기능을 AWS 지원하는지 여부를 알아보려면 섹션을 참조하세요IAM AWS 서비스 작업 방식.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유 AWS 계정 한 다른의 IAM 사용자에게 액세스 권한 제공을 참조하세요.
- 리소스에 대한 액세스 권한을 타사에 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정 소유에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

이 AWS 제품 또는 서비스에 대한 규정 준수 검증

AWS 서비스 가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 규정 준수 AWS 서비스 프로그램 범위규정 준수 섹션을 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 AWS 규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 Downloading Reports inDownloading AWS Artifact 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 AWS 서비스 결정됩니다.는 규정 준수를 지원하기 위해 다음 리소스를 AWS 제공합니다.

- 보안 규정 준수 및 거버넌스 - 이러한 솔루션 구현 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수 기능을 배포하는 단계를 제공합니다.
- HIPAA 적격 서비스 참조-HIPAA 적격 서비스가 나열되어 있습니다. 모든가 HIPAA에 적합한 AWS 서비스 것은 아닙니다.
- AWS 규정 준수 리소스 -이 워크북 및 가이드 모음은 업계 및 위치에 적용될 수 있습니다.
- AWS 고객 규정 준수 가이드 - 규정 준수의 관점에서 공동 책임 모델을 이해합니다. 이 가이드에는 여러 프레임워크(미국 국립표준기술연구소(NIST), 결제카드 산업 보안 표준 위원회(PCI), 국제표준화기구(ISO))의 보안 제어에 대한 지침을 보호하고 AWS 서비스 매핑하는 모범 사례가 요약되어 있습니다.
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 -이 AWS Config 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.

- AWS Security Hub - 이를 AWS 서비스 통해 내 보안 상태를 포괄적으로 볼 수 있습니다 AWS. Security Hub는 보안 컨트롤을 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하세요.
- Amazon GuardDuty - 의심스러운 악의적인 활동이 있는지 환경을 모니터링하여 사용자, AWS 계정 워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 AWS 서비스 탐지합니다. GuardDuty는 특정 규정 준수 프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하여 PCI DSS와 같은 다양한 규정 준수 요구 사항을 따르는 데 도움을 줄 수 있습니다.
- AWS Audit Manager - 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 위험과 규정 및 업계 표준 준수를 관리하는 방법을 간소화할 수 있습니다.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 AWS 서비스 보안 설명서 페이지 및 규정 AWSAWS 준수 프로그램의 규정 준수 노력 범위에 속하는 서비스를 참조하세요.

이 AWS 제품 또는 서비스에 대한 복원력

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 기반으로 구축됩니다.

AWS 리전 는 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결된 물리적으로 분리되고 격리된 여러 가용 영역을 제공합니다.

가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 AWS 서비스 보안 설명서 페이지 및 규정 AWSAWS 준수 프로그램의 규정 준수 노력 범위에 속하는 서비스를 참조하세요.

이 AWS 제품 또는 서비스에 대한 인프라 보안

이 AWS 제품 또는 서비스는 관리형 서비스를 사용하므로 글로벌 네트워크 보안으로 AWS 보호됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 Security Pillar AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS 게시된 API 호출을 사용하여 네트워크를 통해이 AWS 제품 또는 서비스에 액세스합니다. 고객은 다음을 지원해야 합니다.

- Transport Layer Security(TLS) TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군 Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 위탁자와 관련된 보안 암호 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 자격 증명을 생성하여 요청에 서명할 수 있습니다.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 AWS 서비스 보안 설명서 페이지 및 규정 AWSAWS 준수 프로그램의 규정 준수 노력 범위에 속하는 서비스를 참조하세요.

문서 기록

이 주제에서는 기록 과정에서 AWS SDK for Kotlin 개발자 안내서의 중요한 변경 사항에 대해 설명합니다.

변경 사항	설명	날짜
체크섬을 통한 데이터 무결성 보호	자동 체크섬 계산에 대한 세부 정보로 콘텐츠가 업데이트되었습니다.	2025년 1월 16일
기본 자격 증명 공급자 체인 콘텐츠 업데이트	기본 자격 증명 공급자 체인입니다.	2025년 1월 15일
빌드 파일 업데이트 예제	Gradle 버전 8.11.1에서 생성된 빌드 파일 요소를 표시합니다. BOM 사용을 표시합니다. 최신 버전의 아티팩트에 대한 링크를 포함합니다.	2024년 12월 18일
DynamoDB Mapper(개발자 미리 보기) 주제 추가	DynamoDB Mapper(개발자 미리 보기)를 사용하여 클래스를 DynamoDB 항목에 매핑	2024년 10월 29일
Amazon S3 버킷 이름 업데이트	를 사용한 Amazon S3 체크섬 AWS SDK for Kotlin	2024년 9월 30일
OkHttp4 엔진 정보 추가	HTTP 엔진 유형 지정	2024년 9월 26일
DynamoDB의 AWS 계정 기반 엔드포인트에 대한 정보 추가	AWS 계정 기반 엔드포인트 사용	2024년 9월 24일
문제 해결 FAQs 주제 추가	문제 해결 FAQ	2024년 9월 18일
OpenTelemetry 구성 예제 및 기본 글로벌 원격 측정 공급자의 구성 업데이트	Observability	2024년 5월 2일

서비스 클라이언트 생성 프로세스에 대한 세부 정보 제공	서비스 클라이언트 생성	2024년 3월 14일
다중 리전 액세스 포인트 주제 추가	SDK for Kotlin을 사용하여 Amazon S3 다중 리전 액세스 포인트 작업	2024년 2월 6일
Gradle 버전 카탈로그 지침 추가	Gradle 버전 카탈로그(탭)	2023년 12월 19일
정식 출시 릴리스	AWS SDK for Kotlin 개발자 안내서	2023년 11월 27일
SDK 업데이트를 기반으로 클라이언트 엔드포인트 구성 섹션 업데이트	클라이언트 엔드포인트	2023년 8월 25일
아마존 S3 체크섬	Amazon S3에서 유연한 체크섬을 사용하는 방법에 대한 섹션이 추가되었습니다.	2023년 8월 14일
관찰성 주제 추가	Observability	2023년 8월 3일
재시도를 설명하는 주제 추가	재시도	2023년 7월 7일
SDK 업데이트를 기반으로 HTTP 클라이언트 구성 섹션 업데이트	HTTP 클라이언트 구성	2023년 6월 6일
HTTP 사전 서명 주제 추가	요청 사전 서명	2023년 6월 2일
HTTP 인터셉터 주제 추가	HTTP 인터셉터	2023년 5월 22일
자동 토큰 새로 고침 지원	Single Sign-On 액세스에 대한 지침을 업데이트합니다.	2023년 5월 18일
아마존 S3 체크섬	Amazon S3에서 체크섬을 사용하는 방법을 설명하는 섹션을 추가합니다.	2023년 5월 15일

서비스 클라이언트 구성 재정의	서비스 클라이언트의 구성을 재정의하는 방법과 리소스가 영향을 받는 방법을 설명하는 섹션을 추가합니다.	2023년 5월 8일
최소 TLS 버전 적용	최소 TLS 버전을 적용하는 옵션을 설명하는 섹션을 추가합니다.	2023년 5월 3일
클라이언트 엔드포인트 구성	클라이언트 엔드포인트 구성에 대해 설명하는 주제를 추가합니다.	2023년 4월 7일
IAM 모범 사례 업데이트	IAM 모범 사례에 따라 가이드가 업데이트되었습니다. 자세한 내용은 IAM의 보안 모범 사례를 참조하세요.	2023년 3월 22일
예제 Maven 프로젝트 파일 추가	설정 주제의 Gradle의 프로젝트 파일 외에도 Maven 프로젝트 파일의 예를 보여줍니다.	2022년 12월 2일
개발자 안내서 미리 보기의 콘텐츠 수정	최근 개발 작업을 반영하도록 콘텐츠 업데이트	2022년 10월 4일
AWS SDK for Kotlin 개발자 미리 보기 릴리스	AWS SDK for Kotlin	2021년 12월 2일
AWS SDK for Kotlin 알파 릴리스	새 AWS SDK for Kotlin 알파 릴리스 발표	2021년 8월 30일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.