

Panduan Developer

# AWS SDK for Kotlin



# AWS SDK for Kotlin: Panduan Developer

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

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang mungkin menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang menghina atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

---

# Table of Contents

Apa itu AWS SDK for Kotlin? .....	1
Memulai dengan SDK .....	1
Pemeliharaan dan dukungan untuk versi SDK utama .....	1
Sumber daya tambahan .....	1
Memulai .....	3
Langkah 1: Siapkan untuk tutorial ini .....	3
Langkah 2: Buat proyek .....	3
Langkah 3: Tulis kodenya .....	6
Langkah 4: Bangun dan jalankan aplikasi .....	8
Berhasil .....	9
Pembersihan .....	9
Langkah selanjutnya .....	9
Penyiapan .....	10
Pengaturan dasar .....	10
Gambaran Umum .....	10
Kemampuan masuk ke portal AWS akses .....	11
Konfigurasi sistem masuk tunggal .....	12
Masuk menggunakan AWS CLI .....	12
Instal Java dan alat build .....	13
Gunakan kredensial sementara .....	14
Buat file build proyek .....	15
Kode proyek Anda .....	20
Masuk menggunakan AWS CLI .....	20
Konfigurasi .....	22
Buat klien layanan .....	24
Konfigurasi klien dalam kode .....	24
Konfigurasi klien dari lingkungan .....	25
Tutup klien .....	26
Wilayah AWS seleksi .....	27
Rantai penyedia Wilayah default .....	27
Penyedia kredensial .....	28
Rantai penyedia kredensial default .....	28
Penyedia kredensial eksplisit .....	31
Titik akhir klien .....	31

Konfigurasi kustom .....	32
Contoh .....	35
HTTP .....	36
Konfigurasi klien HTTP .....	36
Gunakan proxy HTTP .....	40
Pencegat .....	40
Menerapkan versi TLS minimum .....	42
Percobaan ulang .....	44
Konfigurasi default .....	44
Upaya maksimal .....	44
Penundaan dan backoff .....	45
Coba lagi ember token .....	48
Coba ulang adaptif .....	51
Observabilitas .....	52
Konfigurasi a TelemetryProvider .....	53
Metrik .....	54
Pencatatan log .....	57
Penyedia telemetri .....	60
Ganti konfigurasi klien .....	62
Siklus hidup klien yang diganti .....	63
Sumber Daya Bersama .....	63
Gunakan SDK .....	65
Membuat permintaan .....	65
DSLKelebihan antarmuka layanan .....	66
Permintaan tanpa input yang diperlukan .....	67
Coroutine .....	67
Membuat permintaan bersamaan .....	67
Membuat permintaan pemblokiran .....	68
Operasi streaming .....	69
Respons streaming .....	69
Permintaan streaming .....	70
Paginasi .....	71
Pelayan .....	72
Penanganan kesalahan .....	73
Pengecualian layanan .....	73
Pengecualian klien .....	74

Metadata kesalahan .....	74
Permintaan presign .....	75
Dasar-dasar pra-penandatanganan .....	75
Konfigurasi presigning lanjutan .....	76
Presigning POST dan permintaan PUT .....	76
Operasi yang SDK dapat dilakukan sebelumnya .....	77
Pemecahan Masalah FAQs .....	78
Bagaimana cara memperbaiki masalah “koneksi ditutup”? .....	78
Mengapa pengecualian dilemparkan sebelum mencapai upaya maksimal? .....	79
Bagaimana cara memperbaiki <code>NoSuchMethodError</code> atau <code>NoClassDefFoundError</code> ? .....	80
Bekerja dengan Layanan AWS .....	82
Amazon S3 .....	83
Perlindungan integritas data dengan checksum .....	84
Bekerja dengan Titik Akses Multi-Wilayah .....	88
DynamoDB .....	94
Gunakan AWS titik akhir berbasis akun .....	94
Gunakan DynamoDB Mapper (Pratinjau Pengembang) .....	95
Contoh kode .....	122
API Gateway .....	123
Skenario .....	124
Aurora .....	124
Hal-hal mendasar .....	125
Tindakan .....	138
Skenario .....	124
Auto Scaling .....	152
Hal-hal mendasar .....	125
Tindakan .....	138
Amazon Bedrock .....	169
Tindakan .....	138
CloudWatch .....	170
Hal-hal mendasar .....	125
Tindakan .....	138
CloudWatch Log .....	210
Tindakan .....	138
Penyedia Identitas Amazon Cognito .....	214
Tindakan .....	138

Skenario .....	124
Amazon Comprehend .....	229
Skenario .....	124
DynamoDB .....	230
Hal-hal mendasar .....	125
Tindakan .....	138
Skenario .....	124
Amazon EC2 .....	261
Hal-hal mendasar .....	125
Tindakan .....	138
Amazon ECR .....	291
Hal-hal mendasar .....	125
Tindakan .....	138
OpenSearch Layanan .....	320
Tindakan .....	138
EventBridge .....	324
Hal-hal mendasar .....	125
Tindakan .....	138
AWS Glue .....	354
Hal-hal mendasar .....	125
Tindakan .....	138
IAM .....	366
Hal-hal mendasar .....	125
Tindakan .....	138
AWS IoT .....	385
Hal-hal mendasar .....	125
Tindakan .....	138
AWS IoT data .....	409
Tindakan .....	138
Amazon Keyspaces .....	411
Hal-hal mendasar .....	125
Tindakan .....	138
AWS KMS .....	436
Tindakan .....	138
Lambda .....	446
Hal-hal mendasar .....	125

Tindakan .....	138
Skenario .....	124
MediaConvert .....	455
Tindakan .....	138
Amazon Pinpoint .....	469
Tindakan .....	138
Amazon RDS .....	479
Hal-hal mendasar .....	125
Tindakan .....	138
Skenario .....	124
Layanan Data Amazon RDS .....	497
Skenario .....	124
Amazon Redshift .....	498
Tindakan .....	138
Skenario .....	124
Amazon Rekognition .....	502
Tindakan .....	138
Skenario .....	124
Registrasi domain Route 53 .....	521
Hal-hal mendasar .....	125
Tindakan .....	138
Amazon S3 .....	540
Hal-hal mendasar .....	125
Tindakan .....	138
Skenario .....	124
SageMaker AI .....	562
Tindakan .....	138
Skenario .....	124
Secrets Manager .....	587
Tindakan .....	138
Amazon SES .....	588
Skenario .....	124
Amazon SNS .....	590
Tindakan .....	138
Skenario .....	124
Amazon SQS .....	618

Tindakan .....	138
Skenario .....	124
Step Functions .....	640
Hal-hal mendasar .....	125
Tindakan .....	138
Dukungan .....	661
Hal-hal mendasar .....	125
Tindakan .....	138
Amazon Translate .....	679
Skenario .....	124
Keamanan .....	681
Perlindungan data .....	681
Menegakkan TLS 1.2 .....	683
Dukungan TLS di Jawa .....	683
Cara memeriksa versi TLS .....	683
Identity and Access Management .....	683
Audiens .....	684
Mengautentikasi dengan identitas .....	684
Mengelola akses menggunakan kebijakan .....	688
Bagaimana Layanan AWS bekerja dengan IAM .....	691
Memecahkan masalah AWS identitas dan akses .....	691
Validasi Kepatuhan .....	693
Ketahanan .....	694
Keamanan Infrastruktur .....	695
Riwayat dokumen .....	696
.....	dcc



# Apa itu AWS SDK for Kotlin?

AWS SDK for Kotlin Ini menyediakan Kotlin APIs untuk Amazon Web Services. Dengan menggunakan iniSDK, Anda dapat membuat aplikasi Kotlin yang berfungsi dengan Amazon S3, Amazon, EC2 Amazon DynamoDB, dan lainnya. Dengan KotlinSDK, Anda dapat menargetkan JVM platform atau Android API level 24 atau lebih tinggi. Support untuk platform tambahan seperti JavaScript dan Native akan hadir di rilis mendatang.

Untuk melacak fitur yang akan datang di rilis mendatang, lihat [peta jalan kami di GitHub](#).

## Memulai dengan SDK

Untuk memulaiSDK, ikuti [Memulai](#) tutorialnya.

Untuk mengatur lingkungan pengembangan Anda, lihat[Penyiapan](#).

Untuk membuat dan mengonfigurasi klien layanan untuk membuat permintaan Layanan AWS, lihat [Konfigurasi](#). Untuk informasi tentang berbagai fiturSDK, lihat[Gunakan SDK](#).

Untuk kasus penggunaan dan contoh melakukan API operasi tertentu, lihat[Contoh kode](#).

## Pemeliharaan dan dukungan untuk versi SDK utama

Untuk informasi tentang pemeliharaan dan dukungan untuk versi SDK utama dan dependensi yang mendasarinya, lihat topik berikut di Panduan Referensi Alat AWS SDKs dan Alat:

- [AWS SDKsdan Kebijakan Pemeliharaan Alat](#)
- [AWS SDKsdan Tools Version Support Matrix](#)

## Sumber daya tambahan

Selain panduan ini, berikut ini adalah sumber daya online yang berharga SDK untuk pengembang Kotlin:

- [AWS blog pengembang](#)
- [Forum pengembang](#)
- [SDKsumber](#) (GitHub)

- [Katalog Kode Sampel AWS](#)
- [@awsdevelopers](#) (X, sebelumnya Twitter)

# Memulai SDK untuk Kotlin

AWS SDK for Kotlin Menyediakan Kotlin APIs untuk masing-masing Layanan AWS. Dengan menggunakan SDK, Anda dapat membuat aplikasi Kotlin yang berfungsi dengan Amazon S3, Amazon, EC2 Amazon DynamoDB, dan lainnya.

Tutorial ini menunjukkan cara menggunakan Gradle untuk mendefinisikan dependensi untuk AWS SDK for Kotlin Kemudian, Anda membuat kode yang menulis data ke tabel DynamoDB. Meskipun Anda mungkin ingin menggunakan fitur IDE, yang Anda butuhkan untuk tutorial ini adalah jendela terminal dan editor teks.

Ikuti langkah-langkah berikut untuk menyelesaikan tutorial ini:

- [Langkah 1: Siapkan tutorial ini](#)
- [Langkah 2: Buat proyek](#)
- [Langkah 3: Tulis kodenya](#)
- [Langkah 4: Bangun dan jalankan aplikasi](#)

## Langkah 1: Siapkan untuk tutorial ini

Sebelum memulai tutorial ini, Anda memerlukan [set izin IAM Identity Center](#) yang dapat mengakses DynamoDB dan Anda memerlukan lingkungan pengembangan Kotlin yang dikonfigurasi dengan pengaturan masuk tunggal IAM Identity Center untuk mengaksesnya. AWS

Ikuti petunjuk dalam [Pengaturan dasar](#) panduan ini untuk mendapatkan pengaturan dasar untuk tutorial ini.

Setelah Anda mengonfigurasi lingkungan pengembangan Anda dengan [akses masuk tunggal](#) untuk Kotlin SDK dan Anda memiliki [sesi portal AWS akses aktif](#), lanjutkan dengan Langkah 2.

## Langkah 2: Buat proyek

Untuk membuat proyek untuk tutorial ini, pertama-tama gunakan Gradle untuk membuat file dasar untuk proyek Kotlin. Kemudian, perbarui file dengan pengaturan, dependensi, dan kode yang diperlukan untuk file. AWS SDK for Kotlin

Untuk membuat proyek baru menggunakan Gradle

**Note**

Tutorial ini menggunakan Gradle versi 8.11.1 dengan `gradle init` perintah, yang menawarkan lima petunjuk pada langkah 3 di bawah ini. Jika Anda menggunakan versi Gradle yang berbeda, petunjuknya mungkin berbeda dan juga versi artefak yang telah diisi sebelumnya.

1. Buat direktori baru yang disebut `getstarted` di lokasi pilihan Anda, seperti desktop atau folder rumah Anda.
2. Buka terminal atau jendela prompt perintah dan arahkan ke `getstarted` direktori yang Anda buat.
3. Gunakan perintah berikut untuk membuat proyek Gradle baru dan class Kotlin dasar.

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

- Saat diminta untuk `targetJava version`, tekan Enter (default ke). 21
- Ketika diminta dengan `Project name`, tekan Enter (default ke nama direktori, `getstarted` dalam tutorial ini).
- Saat diminta `application structure`, tekan Enter (default ke). `Single application project`
- Saat diminta `Select test framework`, tekan Enter (default ke). `kotlin.test`
- Saat diminta `Generate build using new APIs and behavior`, tekan Enter (default ke). `no`

Untuk mengonfigurasi proyek Anda dengan dependensi untuk AWS SDK for Kotlin dan Amazon S3

- Di `getstarted` direktori yang Anda buat pada prosedur sebelumnya, ganti isi `settings.gradle.kts` file dengan konten berikut, ganti `X.Y.Z` dengan SDK [versi terbaru](#) untuk Kotlin:

```
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")
```

- Arahkan ke gradle direktori di dalam getstarted direktori. Ganti isi file katalog versi bernama `libs.versions.toml` dengan konten berikut:

```
[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" }
```

- Arahkan ke app direktori dan buka `build.gradle.kts` file. Ganti isinya dengan kode berikut, lalu simpan perubahan Anda.

```
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()
}
```

dependenciesBagian ini berisi implementation entri untuk modul Amazon S3 dari AWS SDK for Kotlin Kompiler Gradle dikonfigurasi untuk menggunakan Java 21 di bagian tersebut. java

## Langkah 3: Tulis kodenya

Setelah proyek dibuat dan dikonfigurasi, edit kelas default proyek App untuk menggunakan kode contoh berikut.

1. Di folder proyek Andaapp, navigasikan ke direktorisrc/main/kotlin/org/example. Buka file App.kt.
2. Ganti isinya dengan kode berikut dan simpan file.

```
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 = ByteString.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!")  
}
```

## Langkah 4: Bangun dan jalankan aplikasi

Setelah proyek dibuat dan berisi kelas contoh, bangun dan jalankan aplikasi.

1. Buka terminal atau jendela prompt perintah dan arahkan ke direktori proyek `Andagetstarted`.
2. Gunakan perintah berikut untuk membangun dan menjalankan aplikasi Anda:

```
gradle run
```

### Note

Jika Anda mendapatkan `IdentityProviderException`, Anda mungkin tidak memiliki sesi masuk tunggal yang aktif. Jalankan perintah `aws sso login` AWS CLI untuk memulai sesi baru.

Aplikasi memanggil operasi [CreateBucket](#) API untuk membuat bucket S3 baru dan kemudian memanggil [putObject](#) untuk memasukkan objek baru ke dalam bucket S3 baru.

Dalam `cleanup()` fungsi di akhir, aplikasi menghapus objek dan kemudian menghapus bucket S3.

Untuk melihat hasilnya di konsol Amazon S3

1. Masuk `App.kt`, komentari baris `cleanup(s3)` di `runBlocking` bagian dan simpan file.
2. Bangun kembali proyek dan masukkan objek baru ke dalam bucket S3 baru dengan menjalankannya `gradle run`
3. Masuk ke [konsol Amazon S3](#) untuk melihat objek baru di bucket S3 baru.

Setelah Anda melihat objek, hapus ember S3.



## Berhasil

Jika proyek Gradle Anda dibangun dan dijalankan tanpa kesalahan, maka selamat. Anda telah berhasil membangun aplikasi Kotlin pertama Anda menggunakan AWS SDK for Kotlin

## Pembersihan

Setelah Anda selesai mengembangkan aplikasi baru Anda, hapus AWS sumber daya apa pun yang Anda buat selama tutorial ini untuk menghindari biaya apa pun. Anda mungkin juga ingin menghapus atau mengarsipkan folder proyek (`get-started`) yang Anda buat di Langkah 2.

Ikuti langkah-langkah ini untuk membersihkan sumber daya:

- Jika Anda mengomentari panggilan ke `cleanup()` fungsi tersebut, hapus bucket S3 dengan menggunakan konsol [Amazon S3](#).

## Langkah selanjutnya

Sekarang setelah Anda memiliki dasar-dasarnya, Anda dapat mempelajari hal-hal berikut:

- [Langkah-langkah penyiapan tambahan untuk bekerja dengan SDK untuk Kotlin](#)
- [Konfigurasi SDK untuk Kotlin](#)
- [Menggunakan SDK untuk Kotlin](#)
- [Keamanan untuk SDK untuk Kotlin](#)

# Mengatur AWS SDK for Kotlin

Untuk membuat permintaan untuk Layanan AWS menggunakan AWS SDK for Kotlin, Anda memerlukan yang berikut:

- Kemampuan untuk masuk ke portal AWS akses
- Izin untuk menggunakan AWS sumber daya yang dibutuhkan aplikasi Anda
- Lingkungan pengembangan dengan elemen-elemen berikut:
  - [File konfigurasi bersama](#) yang diatur dengan setidaknya salah satu cara berikut:
    - `configFile` berisi setelan kredensial Pusat Identitas IAM sehingga SDK dapat memperoleh kredensial AWS
    - `credentialsFile` berisi kredensi sementara
  - [Alat otomatisasi build seperti Gradle atau Maven](#)
- Sesi portal AWS akses aktif saat Anda siap menjalankan aplikasi

Dalam topik ini:

- [Pengaturan dasar](#)
- [Buat file build proyek](#)
- [Kode project Kotlin Anda menggunakan SDK untuk Kotlin](#)

## Pengaturan dasar

### Gambaran Umum

Agar berhasil mengembangkan aplikasi yang mengakses Layanan AWS menggunakan AWS SDK for Kotlin, persyaratan berikut harus dipenuhi.

- Anda harus dapat [masuk ke portal AWS akses](#) yang tersedia di AWS IAM Identity Center.
- [Izin peran IAM yang](#) dikonfigurasi untuk SDK harus mengizinkan akses ke Layanan AWS yang dibutuhkan aplikasi Anda. Izin yang terkait dengan kebijakan `PowerUserAccess` AWS terkelola cukup untuk sebagian besar kebutuhan pengembangan.
- Lingkungan pengembangan dengan elemen-elemen berikut:
  - [File konfigurasi bersama](#) yang diatur setidaknya dalam salah satu cara berikut:

- `configFile` ini berisi [pengaturan masuk tunggal Pusat Identitas IAM](#) sehingga SDK bisa mendapatkan kredensi. AWS
- `credentialsFile` tersebut berisi kredensi sementara.
- [Instalasi Java 8 atau yang lebih baru.](#)
- [Alat otomatisasi build seperti Maven atau Gradle.](#)
- Editor teks untuk bekerja dengan kode.
- [\(Opsional, tetapi disarankan\) IDE \(lingkungan pengembangan terintegrasi\) seperti IntelliJ IDEA atau Eclipse.](#)

Saat Anda menggunakan IDE, Anda juga dapat mengintegrasikan AWS Toolkit s agar lebih mudah digunakan Layanan AWS. [AWS Toolkit for Eclipse](#)Ini [AWS Toolkit for IntelliJ](#)dan dua toolkit yang dapat Anda gunakan.

- Sesi portal AWS akses aktif saat Anda siap menjalankan aplikasi Anda. Anda menggunakan AWS Command Line Interface untuk [memulai proses masuk ke portal akses](#) IAM Identity Center. AWS

#### Important

Petunjuk di bagian penyiapan ini mengasumsikan bahwa Anda atau organisasi menggunakan IAM Identity Center. Jika organisasi Anda menggunakan penyedia identitas eksternal yang bekerja secara independen dari IAM Identity Center, cari tahu cara mendapatkan kredensi sementara untuk SDK untuk Kotlin untuk digunakan. Ikuti petunjuk ini untuk menambahkan kredensi sementara ke file. `~/.aws/credentials`

Jika penyedia identitas Anda menambahkan kredensi sementara secara otomatis ke `~/.aws/credentials` file, pastikan bahwa nama profil tersebut `[default]` sehingga Anda tidak perlu memberikan nama profil ke SDK atau. AWS CLI

## Kemampuan masuk ke portal AWS akses

Portal AWS akses adalah lokasi web tempat Anda masuk secara manual ke Pusat Identitas IAM. Format URL adalah `d-xxxxxxxxx.awsapps.com/start` atau `your_subdomain.awsapps.com/start`.

Jika Anda tidak terbiasa dengan portal AWS akses, ikuti panduan untuk akses akun di topik [otentikasi Pusat Identitas IAM](#) di Panduan Referensi AWS SDKs dan Alat.

## Menyiapkan akses masuk tunggal untuk SDK

Setelah Anda menyelesaikan Langkah 2 di [bagian akses terprogram](#) agar SDK menggunakan autentikasi IAM Identity Center, sistem Anda harus berisi elemen-elemen berikut.

- Itu AWS CLI, yang Anda gunakan untuk memulai [sesi portal AWS akses](#) sebelum Anda menjalankan aplikasi Anda.
- `~/.aws/configFile` yang berisi [profil default](#). SDK untuk Kotlin menggunakan konfigurasi penyedia token SSO profil untuk memperoleh kredensial sebelum mengirim permintaan ke `AWSsso_role_name` Nilai, yang merupakan peran IAM yang terhubung ke set izin Pusat Identitas IAM, harus memungkinkan akses ke yang Layanan AWS digunakan dalam aplikasi Anda.

`configFile` contoh berikut menunjukkan profil default yang diatur dengan konfigurasi penyedia token SSO. `sso_session` Pengaturan profil mengacu pada `sso-session` bagian bernama. `sso-session` Bagian ini berisi pengaturan untuk memulai sesi portal AWS akses.

```
[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
```

Untuk detail selengkapnya tentang pengaturan yang digunakan dalam konfigurasi penyedia token SSO, lihat [konfigurasi penyedia token SSO](#) di Panduan Referensi Alat AWS SDKs dan Alat.

Jika lingkungan pengembangan Anda tidak diatur untuk akses terprogram seperti yang ditunjukkan sebelumnya, ikuti [Langkah 2 di Panduan SDKs Referensi](#).

## Masuk menggunakan AWS CLI

Sebelum menjalankan aplikasi yang mengakses Layanan AWS, Anda memerlukan sesi portal AWS akses aktif agar SDK menggunakan autentikasi IAM Identity Center untuk menyelesaikan kredensialnya. Jalankan perintah berikut di AWS CLI untuk masuk ke portal AWS akses.

```
aws sso login
```

Karena Anda memiliki pengaturan profil default, Anda tidak perlu memanggil perintah dengan `--profile` opsi. Jika konfigurasi penyedia token SSO Anda menggunakan profil bernama, perintahnya adalah `aws sso login --profile named-profile`.

Untuk menguji apakah Anda sudah memiliki sesi aktif, jalankan AWS CLI perintah berikut.

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

Respons terhadap perintah ini harus melaporkan akun IAM Identity Center dan set izin yang dikonfigurasi dalam config file bersama.

#### Note

Jika Anda sudah memiliki sesi portal AWS akses aktif dan menjalankan `aws sso login`, Anda tidak akan diminta untuk memberikan kredensi. Namun, Anda akan melihat dialog yang meminta izin botocore untuk mengakses informasi Anda. botocore adalah fondasi untuk AWS CLI. Pilih Izinkan untuk mengotorisasi akses ke informasi Anda untuk AWS CLI dan SDK untuk Kotlin.

## Instal Java dan alat build

Lingkungan pengembangan Anda membutuhkan yang berikut:

- JDK 8 atau yang lebih baru. [Ini AWS SDK for Kotlin bekerja dengan Oracle Java SE Development Kit dan dengan distribusi Open Java Development Kit \(OpenJDK\) seperti, Red Hat OpenJDK, Amazon Corretto dan JDK. AdoptOpen](#)
- Alat build atau IDE yang mendukung Maven Central seperti Apache Maven, Gradle, atau IntelliJ.
  - [Untuk informasi tentang cara menginstal dan menggunakan Maven, lihat http://maven.apache.org/.](http://maven.apache.org/)
  - Untuk informasi tentang cara menginstal dan menggunakan Gradle, lihat [https://gradle.org/.](https://gradle.org/)
  - Untuk informasi tentang cara menginstal dan menggunakan IntelliJ IDEA, lihat. <https://www.jetbrains.com/idea/>



SDK untuk Kotlin akan mengakses kredensi sementara ini saat membuat klien layanan dan menggunakannya untuk setiap permintaan. Pengaturan untuk peran IAM yang dipilih pada langkah 5a menentukan [berapa lama kredensi sementara valid](#). Durasi maksimum adalah dua belas jam.

Setelah kredensi sementara kedaluwarsa, ulangi langkah 4 hingga 7.

## Buat file build proyek

Setelah mengonfigurasi akses masuk tunggal dan lingkungan pengembangan, buat project Kotlin menggunakan alat build pilihan Anda. Dalam file build, tentukan dependensi Layanan AWS yang perlu diakses aplikasi Anda.

Untuk melihat daftar semua kemungkinan nama artefak Maven, lihat dokumentasi referensi [API](#).

Untuk menemukan versi terbaru SDK, periksa [rilis terbaru di GitHub](#).

Contoh file build berikut menyediakan elemen yang diperlukan untuk memulai pengkodean proyek dengan tujuh Layanan AWS.

### Gradle

Ini AWS SDK for Kotlin menerbitkan [katalog versi Gradle](#) dan bill of materials (BOM) yang dapat membantu Anda menemukan nama dependensi dan menjaga nomor versi tetap disinkronkan di beberapa artefak.

Perhatikan bahwa katalog versi adalah fitur pratinjau Gradle sebelum versi 8. Bergantung pada versi Gradle yang Anda gunakan, Anda mungkin perlu ikut serta melalui API [Pratinjau Fitur](#).

Untuk menggunakan katalog versi Gradle

1. Dalam `settings.gradle.kts` file Anda, tambahkan `versionCatalogs` blok di dalam `dependencyResolutionManagement` blok.

Contoh file berikut mengkonfigurasi katalog AWS SDK for Kotlin versi. Anda dapat menavigasi ke `X.Y.Z` tautan untuk melihat versi terbaru yang tersedia.

```
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. Deklarasikan dependensi `build.gradle.kts` dengan menggunakan pengidentifikasi type-safe yang disediakan oleh katalog versi.

Contoh file berikut mendeklarasikan dependensi untuk tujuh. Layanan AWS

```
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"
}
```

\* Versi Java, misalnya 17 atau 21.

## Maven

Contoh pom.xml file berikut memiliki dependensi untuk tujuh. Layanan AWS Anda dapat menavigasi ke [X.Y.Z](#) tautan untuk melihat versi terbaru yang tersedia.

```
<?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>
            </executions>
        </plugin>
    </plugins>
</build>
```

```
        <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>
```

\* Versi Java, misalnya 17 atau 21.

## Kode project Kotlin Anda menggunakan SDK untuk Kotlin

Sekarang kesenangan dimulai. Saat Anda mengembangkan aplikasi, Anda dapat merujuk ke [Referensi AWS SDK for Kotlin API](#) untuk informasi lengkap tentang operasi API. Gunakan tautan berikut untuk informasi umum Kotlin API:

- [Referensi API perpustakaan standar](#)
- [Ikhtisar Coroutine](#)
- [API Coroutine](#)

## Masuk menggunakan AWS CLI

Setiap kali Anda menjalankan program yang mengakses Layanan AWS, Anda memerlukan sesi portal AWS akses aktif. Anda melakukan ini dengan menjalankan perintah berikut.

```
aws sso login
```

Karena Anda memiliki pengaturan profil default, Anda tidak perlu memanggil perintah dengan `--profile` opsi. Jika konfigurasi masuk tunggal Pusat Identitas IAM Anda menggunakan profil bernama, perintahnya adalah `aws sso login --profile named-profile`

Untuk menguji apakah Anda sudah memiliki sesi aktif, jalankan AWS CLI perintah berikut.

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

Respons terhadap perintah ini harus melaporkan akun IAM Identity Center dan set izin yang dikonfigurasi dalam config file bersama.

# Konfigurasi AWS SDK for Kotlin

Bagian ini menjelaskan cara mengkonfigurasi klien layanan menggunakan AWS SDK for Kotlin. Untuk informasi selengkapnya, lihat [Panduan Referensi SDK dan Alat](#), yang mencakup ikhtisar konfigurasi yang berlaku untuk semua AWS SDKs.

## Daftar Isi

- [Buat klien layanan](#)
  - [Konfigurasi klien dalam kode](#)
  - [Konfigurasi klien dari lingkungan](#)
  - [Tutup klien](#)
- [Wilayah AWS seleksi](#)
  - [Rantai penyedia Wilayah default](#)
- [Penyedia kredensial](#)
  - [Rantai penyedia kredensial default](#)
    - [Pelajari tentang rantai penyedia kredensial default](#)
  - [Penyedia kredensial eksplisit](#)
- [Konfigurasi titik akhir klien](#)
  - [Konfigurasi kustom](#)
    - [Set endpointUrl](#)
    - [Set endpointProvider](#)
      - [EndpointProviderproperty](#)
    - [endpointUrl atau endpointProvider](#)
    - [Catatan tentang Amazon S3](#)
  - [Contoh](#)
    - [Contoh endpointUrl](#)
    - [Contoh endpointProvider](#)
    - [endpointUrl dan endpointProvider](#)
- [HTTP](#)
  - [Konfigurasi klien HTTP](#)
    - [Konfigurasi aturan dasar](#)

- [Impor](#)
- [Kode](#)
- [Tentukan tipe mesin HTTP](#)
  - [Impor](#)
  - [Kode](#)
  - [Gunakan OkHttp4Engine](#)
  - [Gunakan klien HTTP eksplisit](#)
    - [Impor](#)
    - [Kode](#)
- [Gunakan proxy HTTP](#)
  - [Gunakan properti sistem JVM](#)
  - [Menggunakan variabel lingkungan](#)
  - [Gunakan proxy pada EC2 instance](#)
- [Pencegat HTTP](#)
  - [Pendaftaran pencegat](#)
    - [Interceptor untuk semua operasi klien layanan](#)
    - [Pencegat hanya untuk operasi tertentu](#)
- [Menerapkan versi TLS minimum](#)
  - [Konfigurasi mesin HTTP](#)
  - [Mengatur properti sdk.minTls sistem JVM](#)
  - [Mengatur variabel SDK\\_MIN\\_TLS lingkungan](#)
- [Percobaan ulang](#)
  - [Konfigurasi coba lagi default](#)
  - [Upaya maksimal](#)
  - [Penundaan dan backoff](#)
  - [Coba lagi ember token](#)
  - [Coba ulang adaptif](#)
- [Observabilitas](#)
  - [Konfigurasi a TelemetryProvider](#)
    - [Konfigurasi penyedia telemetri global default](#)

- [Konfigurasi penyedia telemetri untuk klien layanan tertentu](#)
- [Metrik](#)
- [Pencatatan log](#)
  - [Tentukan mode log untuk pesan tingkat kabel](#)
    - [Atur mode log dalam kode](#)
    - [Atur mode log dari lingkungan](#)
- [Penyedia telemetri](#)
  - [Konfigurasi penyedia telemetri OpenTelemetry berbasis](#)
    - [Prasyarat](#)
    - [Konfigurasi SDK](#)
    - [Sumber daya](#)
- [Ganti konfigurasi klien layanan](#)
  - [Siklus hidup klien yang diganti](#)
  - [Sumber daya yang dibagikan antar klien](#)

## Buat klien layanan

Untuk membuat permintaan ke Layanan AWS, Anda harus terlebih dahulu membuat instance klien untuk layanan itu.

Anda dapat mengonfigurasi pengaturan umum untuk klien layanan, seperti klien HTTP untuk digunakan, tingkat logging, dan konfigurasi coba lagi. Selain itu, setiap klien layanan memerlukan Wilayah AWS dan penyedia kredensial. SDK menggunakan nilai-nilai ini untuk mengirim permintaan ke Wilayah yang benar dan untuk menandatangani permintaan dengan kredensial yang benar.

Anda dapat menentukan nilai-nilai ini secara terprogram dalam kode atau membuatnya dimuat secara otomatis dari lingkungan.

## Konfigurasi klien dalam kode

Untuk mengonfigurasi klien layanan dengan nilai tertentu, Anda dapat menentukannya dalam fungsi lambda yang diteruskan ke metode pabrik klien layanan seperti yang ditunjukkan pada cuplikan berikut.



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

Nilai apa pun yang tidak Anda tentukan di blok konfigurasi disetel ke default. [Misalnya, jika Anda tidak menentukan penyedia kredensial seperti kode sebelumnya, penyedia kredensial default ke rantai penyedia kredensial default.](#)

#### Warning

Beberapa properti seperti `region` tidak memiliki default. Anda harus menentukannya secara eksplisit di blok konfigurasi saat Anda menggunakan konfigurasi terprogram. Jika SDK tidak dapat menyelesaikan properti, permintaan API mungkin gagal.

## Konfigurasi klien dari lingkungan

Saat membuat klien layanan, SDK dapat memeriksa lokasi di lingkungan eksekusi saat ini untuk menentukan beberapa properti konfigurasi. Lokasi tersebut mencakup [file konfigurasi dan kredensial bersama](#), [variabel lingkungan](#), dan properti sistem [JVM](#). Properti yang tersedia untuk diselesaikan termasuk [AWS Wilayah](#), [strategi coba lagi](#), [mode log](#), dan lainnya. Untuk informasi selengkapnya tentang semua setelan yang dapat diselesaikan SDK dari lingkungan eksekusi, lihat [Panduan Referensi Pengaturan Alat AWS SDKs dan Alat](#).

Untuk membuat klien dengan konfigurasi bersumber lingkungan, gunakan metode statis `suspend fun fromEnvironment()` pada antarmuka klien layanan:

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

Membuat klien dengan cara ini berguna saat berjalan di Amazon EC2 AWS Lambda, atau konteks lain di mana konfigurasi klien layanan tersedia dari lingkungan. Ini memisahkan kode Anda dari lingkungan tempat ia berjalan dan membuatnya lebih mudah untuk menerapkan aplikasi Anda ke beberapa Wilayah tanpa mengubah kode.

Selain itu, Anda dapat mengganti properti tertentu dengan meneruskan blok lambda ke `fromEnvironment`. Contoh berikut memuat beberapa properti konfigurasi dari lingkungan (misalnya, Wilayah) tetapi secara khusus mengganti penyedia kredensial untuk menggunakan kredensial dari profil.

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

SDK menggunakan nilai default untuk properti konfigurasi apa pun yang tidak dapat ditentukan dari setelan terprogram atau dari lingkungan. [Misalnya, jika Anda tidak menentukan penyedia kredensial dalam kode atau dalam setelan lingkungan, penyedia kredensial akan default ke rantai penyedia kredensial default.](#)

#### Warning

Beberapa properti seperti Region tidak memiliki default. Anda harus menentukannya dalam pengaturan lingkungan atau secara eksplisit di blok konfigurasi. Jika SDK tidak dapat menyelesaikan properti, permintaan API mungkin gagal.

#### Note

Meskipun properti terkait kredensial—seperti kunci akses sementara dan konfigurasi SSO—dapat ditemukan di lingkungan eksekusi, nilainya tidak bersumber oleh klien pada saat pembuatan. Sebaliknya, nilai diakses oleh lapisan penyedia kredensial pada setiap permintaan.

## Tutup klien

Saat Anda tidak lagi membutuhkan klien layanan, tutup untuk melepaskan sumber daya apa pun yang digunakannya:

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

Karena klien layanan memperluas [Closeable](#) antarmuka, Anda dapat menggunakan [use](#) ekstensi untuk menutup klien secara otomatis di akhir blok seperti yang ditunjukkan pada cuplikan berikut.

```
DynamoDbClient.fromEnvironment().use { dynamoDbClient ->
```

```
// Invoke several DynamoDB operations.  
}
```

Pada contoh sebelumnya, blok lambda menerima referensi ke klien yang baru saja dibuat. Anda dapat memanggil operasi pada referensi klien ini dan ketika blok selesai — termasuk dengan melempar pengecualian — klien ditutup.

## Wilayah AWS seleksi

Dengan Wilayah AWS, Anda dapat mengakses Layanan AWS yang beroperasi di wilayah geografis tertentu. Ini dapat berguna baik untuk redundansi dan untuk menjaga data dan aplikasi Anda berjalan dekat dengan tempat Anda dan pengguna Anda akan mengaksesnya.

### Rantai penyedia Wilayah default

Saat memuat konfigurasi klien layanan [dari lingkungan](#), proses pencarian berikut digunakan:

1. Wilayah eksplisit apa pun yang disetel pada pembuat.
2. Properti sistem `aws.region` JVM diperiksa. Jika disetel, Wilayah itu digunakan dalam konfigurasi klien.
3. Variabel `AWS_REGION` lingkungan diperiksa. Jika disetel, Wilayah itu digunakan dalam konfigurasi klien.
  - a. Catatan: Variabel lingkungan ini diatur oleh wadah Lambda.
4. SDK memeriksa file konfigurasi AWS bersama. Jika `region` properti disetel untuk profil aktif, SDK akan menggunakannya.
  - a. Variabel `AWS_CONFIG_FILE` lingkungan dapat digunakan untuk menyesuaikan lokasi file konfigurasi bersama.
  - b. Properti sistem `aws.profile` JVM atau variabel `AWS_PROFILE` lingkungan dapat digunakan untuk menyesuaikan profil yang dimuat SDK.
5. SDK mencoba menggunakan Layanan Metadata EC2 Instans Amazon untuk menentukan Wilayah instans yang sedang berjalan. EC2
6. Jika Wilayah masih belum diselesaikan pada saat ini, pembuatan klien gagal dengan pengecualian.

# Penyedia kredensial

**⚠** Urutan di mana rantai penyedia kredensial default menyelesaikan kredensial diubah dengan versi 1.4.0 Untuk detailnya, lihat catatan di bawah ini.

Untuk membuat permintaan ke Amazon Web Services menggunakan AWS SDK for Kotlin, SDK menggunakan kredensial yang ditandatangani secara kriptografi yang dikeluarkan oleh AWS. Untuk memperoleh kredensialnya, SDK dapat menggunakan pengaturan konfigurasi yang terletak di beberapa tempat, misalnya properti sistem JVM, variabel lingkungan, AWS config file bersama dan `filecredentials`, dan metadata instans Amazon. EC2

SDK menggunakan abstraksi penyedia kredensial untuk menyederhanakan proses pengambilan kredensial dari berbagai sumber. SDK berisi [beberapa implementasi penyedia kredensial](#).

Misalnya, Jika konfigurasi yang diambil menyertakan setelan akses masuk tunggal Pusat Identitas IAM dari config file bersama, SDK bekerja dengan Pusat Identitas IAM untuk mengambil kredensial sementara yang digunakan untuk membuat permintaan. Layanan AWS Dengan pendekatan ini untuk memperoleh kredensial, SDK menggunakan penyedia IAM Identity Center (juga dikenal sebagai penyedia kredensial SSO). [Bagian penyiapan](#) panduan ini menjelaskan konfigurasi ini.

Untuk menggunakan penyedia kredensial tertentu, Anda dapat menentukannya saat membuat klien layanan. Atau, Anda dapat menggunakan rantai penyedia kredensial default untuk mencari pengaturan konfigurasi secara otomatis.

## Rantai penyedia kredensial default

Jika tidak ditentukan secara eksplisit pada konstruksi klien, SDK untuk Kotlin menggunakan penyedia kredensial yang secara berurutan memeriksa setiap tempat di mana Anda dapat menyediakan kredensial. Penyedia kredensial default ini diimplementasikan sebagai rantai penyedia kredensial.

Untuk menggunakan rantai default untuk menyediakan kredensial dalam aplikasi Anda, buat klien layanan tanpa menyediakan properti secara eksplisit. `credentialsProvider`

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

Untuk informasi selengkapnya tentang pembuatan klien layanan, lihat [membangun dan mengonfigurasi klien](#).

## Pelajari tentang rantai penyedia kredensial default

Rantai penyedia kredensial default mencari konfigurasi kredensial menggunakan urutan yang telah ditentukan berikut. Ketika pengaturan yang dikonfigurasi memberikan kredensial yang valid, rantai berhenti.

### 1. [AWS kunci akses \(properti sistem JVM\)](#)

SDK mencari properti sistem `aws.accessKeyId`, `aws.secretAccessKey`, dan `aws.sessionToken` JVM.

### 2. [AWS kunci akses \(variabel lingkungan\)](#)

SDK mencoba memuat kredensial dari variabel `AWS_ACCESS_KEY_ID` dan `AWS_SECRET_ACCESS_KEY`, dan `AWS_SESSION_TOKEN` lingkungan.

### 3. [Token identitas web](#)

SDK mencari variabel lingkungan `AWS_WEB_IDENTITY_TOKEN_FILE` dan `AWS_ROLE_ARN` (atau properti `aws.webIdentityTokenFile` sistem JVM dan). `aws.roleArn` Berdasarkan informasi token dan perannya, SDK memperoleh kredensial sementara.

### 4. [Profil dalam file konfigurasi](#)

Pada langkah ini, SDK menggunakan pengaturan yang terkait dengan profil. Secara default, SDK menggunakan `credentials` file bersama `AWS config` dan, tetapi jika variabel `AWS_CONFIG_FILE` lingkungan disetel, SDK menggunakan nilai itu. Jika variabel `AWS_PROFILE` lingkungan (atau properti sistem `aws.profile` JVM) tidak disetel, SDK mencari profil "default", jika tidak maka akan mencari profil yang cocok dengan nilai `AWS_PROFILE`'s

SDK mencari profil berdasarkan konfigurasi yang dijelaskan dalam paragraf sebelumnya dan menggunakan pengaturan yang ditentukan di sana. Jika setelan yang ditemukan oleh SDK berisi campuran setelan untuk pendekatan penyedia kredensial yang berbeda, SDK menggunakan urutan berikut:

- a. [AWS tombol akses \(file konfigurasi\)](#) - SDK menggunakan pengaturan untuk `aws_access_key_id`, `aws_secret_access_key`, dan `aws_session_token`.
- b. [Asumsikan konfigurasi peran](#) - Jika SDK menemukan `role_arn` dan `source_profile` atau `credential_source` pengaturan, SDK mencoba untuk mengambil peran. Jika SDK

menemukan `source_profile` setelah, SDK akan mendapatkan kredensial dari profil lain untuk menerima kredensial sementara untuk peran yang ditentukan oleh `role_arn`. Jika SDK menemukan `credential_source` setelah, SDK akan mendapatkan kredensial dari wadah Amazon ECS, EC2 instans Amazon, atau dari variabel lingkungan tergantung pada nilai setelah `credential_source`. Kemudian menggunakan kredensial tersebut untuk memperoleh kredensial sementara untuk peran tersebut.

Profil harus berisi `source_profile` pengaturan atau `credential_source` pengaturan, tetapi tidak keduanya.

- c. [Konfigurasi token identitas web](#) - Jika SDK menemukan `role_arn` dan `web_identity_token_file` mengatur, ia memperoleh kredensial sementara untuk mengakses AWS sumber daya berdasarkan dan token `role_arn`
- d. [Konfigurasi token SSO](#) - Jika SDK menemukan `sso_session`, `sso_account_id`, `sso_role_name` pengaturan (bersama dengan `sso-session` bagian pendamping dalam file konfigurasi), SDK mengambil kredensial sementara dari layanan Pusat Identitas IAM.
- e. [Konfigurasi SSO lama](#) - Jika SDK menemukan `sso_start_url`,, dan `sso_role_name` pengaturan `sso_region`, `sso_account_id`, SDK mengambil kredensial sementara dari layanan Pusat Identitas IAM.
- f. [Konfigurasi proses](#) - Jika SDK menemukan `credential_process` setelah, SDK menggunakan nilai jalur untuk memanggil proses dan memperoleh kredensial sementara.

## 5. [Kredensial kontainer](#)

SDK mencari variabel lingkungan `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` atau `AWS_CONTAINER_CREDENTIALS_FULL_URI` dan `AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE` atau `AWS_CONTAINER_AUTHORIZATION_TOKEN`. Ini menggunakan nilai-nilai ini untuk memuat kredensial dari titik akhir HTTP yang ditentukan melalui permintaan GET.

## 6. [Kredensial IMDS](#)

SDK mencoba mengambil kredensial dari [Layanan Metadata Instance di titik akhir HTTP default](#) atau yang dikonfigurasi. SDK hanya mendukung [IMDSv2](#).

Jika kredensial masih belum diselesaikan pada saat ini, pembuatan klien gagal dengan pengecualian.

### Catatan: Ubah urutan resolusi kredensyal

Urutan resolusi kredensyal yang dijelaskan di atas adalah terkini untuk 1.4.x+ rilis SDK untuk Kotlin. Sebelum 1.4.0 rilis, item nomor 3 dan 4 diaktifkan dan item 4a saat ini mengikuti item 4f saat ini.

## Penyedia kredensyal eksplisit

Alih-alih menggunakan rantai penyedia default, Anda dapat menentukan penyedia kredensyal tertentu atau rantai kustom (`CredentialsProviderChain`) yang harus digunakan SDK. Misalnya, jika Anda menyetel kredensyal default menggunakan variabel lingkungan, berikan `EnvironmentCredentialsProvider` ke pembuat klien, seperti pada cuplikan kode berikut.

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

### Note

Rantai default menyimpan kredensyal, tetapi penyedia mandiri tidak. Anda dapat membungkus penyedia kredensyal apa pun menggunakan `CachedCredentialsProvider` kelas untuk menghindari pengambilan kredensyal yang tidak perlu pada setiap panggilan API. Penyedia yang di-cache hanya mengambil kredensyal baru ketika yang sekarang kedaluwarsa.

### Note

Anda dapat mengimplementasikan penyedia kredensyal atau rantai penyedia Anda sendiri dengan mengimplementasikan antarmuka `CredentialsProvider`

## Konfigurasi titik akhir klien

Saat AWS SDK for Kotlin memanggil Layanan AWS, salah satu langkah pertamanya adalah menentukan ke mana harus merutekan permintaan. Proses ini dikenal sebagai resolusi titik akhir.

Anda dapat mengonfigurasi resolusi titik akhir untuk SDK saat membuat klien layanan. Konfigurasi default untuk resolusi titik akhir biasanya baik-baik saja, tetapi ada beberapa alasan yang mungkin mengarahkan Anda untuk memodifikasi konfigurasi default. Dua contoh alasannya adalah sebagai berikut:

- Membuat permintaan ke versi prarilis layanan atau untuk penyebaran lokal layanan.
- Akses ke fitur layanan tertentu yang belum dimodelkan dalam SDK.

#### Warning

Resolusi titik akhir adalah topik SDK lanjutan. Jika Anda mengubah pengaturan default, Anda berisiko melanggar kode Anda. Pengaturan default harus berlaku untuk sebagian besar pengguna di lingkungan produksi.

## Konfigurasi kustom

Anda dapat menyesuaikan resolusi titik akhir klien layanan dengan dua properti yang tersedia saat Anda membangun klien:

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

### Set `endpointUrl`

Anda dapat menetapkan nilai `endpointUrl` untuk menunjukkan nama host “dasar” untuk layanan. Nilai ini, bagaimanapun, tidak final karena diteruskan sebagai parameter ke `EndpointProvider` instance klien. `EndpointProvider` implementasi kemudian dapat memeriksa dan berpotensi memodifikasi nilai tersebut untuk menentukan titik akhir.

Sebagai contoh, jika Anda menentukan `endpointUrl` nilai untuk klien Amazon Simple Storage Service (Amazon S3) dan menjalankan `GetObject` operasi, implementasi penyedia endpoint default akan menyuntikkan nama bucket ke dalam nilai nama host.

Dalam praktiknya, pengguna menetapkan `endpointUrl` nilai untuk menunjuk pada contoh pengembangan atau pratinjau layanan.



## Set `endpointProvider`

`EndpointProvider` implementasi klien layanan menentukan resolusi akhir titik akhir. `EndpointProviderAntarmuka` yang ditunjukkan dalam blok kode berikut memperlihatkan `resolveEndpoint` metode.

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

Klien layanan memanggil `resolveEndpoint` metode untuk setiap permintaan. Klien layanan menggunakan `Endpoint` nilai yang dikembalikan oleh penyedia tanpa perubahan lebih lanjut.

### `EndpointProvider` properti

`resolveEndpoint` Metode ini menerima `EndpointParameters` objek khusus layanan yang berisi properti yang digunakan dalam resolusi titik akhir.

Setiap layanan mencakup properti dasar berikut.

Nama	Tipe	Deskripsi
<code>region</code>	String	AWS Wilayah klien
<code>endpoint</code>	String	Sebuah representasi string dari nilai set <code>endpointUrl</code>
<code>useFips</code>	Boolean	Apakah titik akhir FIPS diaktifkan dalam konfigurasi klien
<code>useDualStack</code>	Boolean	Apakah titik akhir dual-stack diaktifkan dalam konfigurasi klien

Layanan dapat menentukan properti tambahan yang diperlukan untuk resolusi. Misalnya, Amazon S3 [S3EndpointParameters](#) menyertakan nama bucket dan juga beberapa pengaturan fitur khusus Amazon S3. Misalnya, `forcePathStyle` properti menentukan apakah pengalamatan host virtual dapat digunakan.

Jika Anda menerapkan penyedia Anda sendiri, Anda tidak perlu membuat instance Anda sendiri. `EndpointParameters` SDK menyediakan properti untuk setiap permintaan dan meneruskannya ke implementasi `resolveEndpoint` Anda.

## `endpointUrl` atau `endpointProvider`

Penting untuk dipahami bahwa dua pernyataan berikut TIDAK menghasilkan klien dengan perilaku resolusi titik akhir yang setara:

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

Pernyataan yang menetapkan `endpointUrl` properti menentukan URL dasar yang diteruskan ke penyedia (default), yang dapat dimodifikasi sebagai bagian dari resolusi titik akhir.

Pernyataan yang menetapkan `endpointProvider` menentukan URL akhir yang `S3Client` digunakan.

Meskipun Anda dapat mengatur kedua properti, dalam banyak kasus yang memerlukan penyesuaian, Anda menyediakan salah satunya. Sebagai pengguna SDK umum, Anda paling sering memberikan `endpointUrl` nilai.

## Catatan tentang Amazon S3

Amazon S3 adalah layanan yang kompleks dengan banyak fitur-fiturnya yang dimodelkan melalui penyesuaian titik akhir yang disesuaikan, seperti hosting virtual bucket. Virtual hosting adalah fitur Amazon S3 di mana nama bucket dimasukkan ke dalam nama host.

Karena itu, kami menyarankan Anda untuk tidak mengganti `EndpointProvider` implementasi di klien layanan Amazon S3. Jika Anda perlu memperluas perilaku resolusinya, mungkin dengan mengirimkan permintaan ke tumpukan pengembangan lokal dengan pertimbangan titik akhir

tambahan, kami sarankan untuk membungkus implementasi default. `endpointProviderContoh` berikut menunjukkan contoh implementasi dari pendekatan ini.

## Contoh

### Contoh `endpointUrl`

Cuplikan kode berikut menunjukkan bagaimana titik akhir layanan umum dapat diganti untuk klien Amazon S3.

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

### Contoh `endpointProvider`

Cuplikan kode berikut menunjukkan cara menyediakan penyedia titik akhir kustom yang membungkus implementasi default untuk Amazon S3.

```
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` dan `endpointProvider`

Contoh program berikut menunjukkan interaksi antara `endpointUrl` dan `endpointProvider` pengaturan. Ini adalah kasus penggunaan lanjutan.

```
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

Bagian ini mencakup konfigurasi pengaturan terkait HTTP di AWS SDK for Kotlin

Topik

- [Konfigurasi klien HTTP](#)
- [Gunakan proxy HTTP](#)
- [Pencegat HTTP](#)
- [Menerapkan versi TLS minimum](#)

## Konfigurasi klien HTTP

Secara default, AWS SDK for Kotlin menggunakan klien HTTP berdasarkan [OkHttp](#). Anda dapat mengganti klien HTTP dan konfigurasinya dengan menyediakan klien yang dikonfigurasi secara eksplisit.

**Note**

Secara default, setiap klien layanan menggunakan salinan sendiri dari klien HTTP. Jika Anda menggunakan beberapa layanan dalam aplikasi Anda, Anda mungkin ingin membangun satu klien HTTP dan membagikannya di semua klien layanan.

## Konfigurasi aturan dasar

Saat Anda mengkonfigurasi klien layanan, Anda dapat mengonfigurasi jenis mesin default. SDK mengelola mesin klien HTTP yang dihasilkan dan secara otomatis menutupnya ketika tidak lagi diperlukan.

Contoh berikut menunjukkan konfigurasi klien HTTP selama inisialisasi klien DynamoDB.

### Impor

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

### Kode

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

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

## Tentukan tipe mesin HTTP

Untuk kasus penggunaan yang lebih maju, Anda dapat meneruskan parameter tambahan `httpClient` yang menentukan jenis mesin. Dengan cara ini, Anda dapat mengatur parameter konfigurasi yang unik untuk jenis mesin tersebut.

Contoh berikut menentukan [OkHttpEngine](#) yang dapat Anda gunakan untuk mengkonfigurasi [maxConcurrencyPerHost](#) properti.

## Impor

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

## Kode

```
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.
}
```

Nilai yang mungkin untuk jenis mesin adalah `OkHttpEngine`, [OkHttp4Engine](#), dan [CrtHttpEngine](#).

Untuk menggunakan parameter konfigurasi khusus untuk mesin HTTP, Anda harus menambahkan mesin sebagai dependensi waktu kompilasi. Untuk `OkHttpEngine`, Anda menambahkan dependensi berikut menggunakan Gradle.

(Anda dapat menavigasi ke [X.Y.Z](#) tautan untuk melihat versi terbaru yang tersedia.)

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

Untuk `CrtHttpEngine`, tambahkan dependensi berikut.

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

## Gunakan **OkHttp4Engine**

Gunakan `OkHttp4Engine` jika Anda tidak dapat menggunakan `defaultOkHttpEngine`. [GitHub Repositori smithy-kotlin](#) memiliki informasi tentang cara Anda mengonfigurasi dan menggunakan `OkHttp4Engine`.

### Gunakan klien HTTP eksplisit

Saat Anda menggunakan klien HTTP eksplisit, Anda bertanggung jawab atas masa pakainya, termasuk menutup saat Anda tidak lagi membutuhkannya. Klien HTTP harus hidup setidaknya selama klien layanan apa pun yang menggunakannya.

Contoh kode berikut menunjukkan kode yang membuat klien HTTP tetap hidup saat aktif. `DynamoDbClient` [use](#) Fungsi ini memastikan klien HTTP menutup dengan benar.

### Impor

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

### Kode

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

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

## Gunakan proxy HTTP

Untuk mengakses AWS melalui server proxy menggunakan AWS SDK for Kotlin, Anda dapat mengkonfigurasi properti sistem JVM atau variabel lingkungan. Jika keduanya disediakan, properti sistem JVM diutamakan.

### Gunakan properti sistem JVM

SDK mencari properti sistem JVM `https.proxyHost`, `https.proxyPort` dan `http.nonProxyHosts` Untuk informasi lebih lanjut tentang properti sistem JVM umum ini, lihat [Jaringan dan Proksi](#) dalam dokumentasi Java.

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

### Menggunakan variabel lingkungan

SDK mencari variabel `https_proxy`, `http_proxy`, dan `no_proxy` lingkungan (dan versi kapitalisasi masing-masing).

```
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
```

### Gunakan proxy pada EC2 instance

Jika Anda mengonfigurasi proxy pada EC2 instance yang diluncurkan dengan peran IAM terlampir, pastikan untuk mengecualikan alamat yang digunakan untuk mengakses metadata [instance](#). Untuk melakukan ini, atur properti sistem `http.nonProxyHosts` JVM atau variabel `no_proxy` lingkungan ke alamat IP dari Layanan Metadata Instance, yaitu. `169.254.169.254` Alamat ini tidak berbeda.

```
export no_proxy=169.254.169.254
```

## Pencegat HTTP

Anda dapat menggunakan pencegat untuk menghubungkan eksekusi permintaan dan tanggapan API. Pencegat adalah mekanisme terbuka di mana SDK memanggil kode yang Anda tulis untuk menyuntikkan perilaku ke dalam siklus hidup permintaan/respons. Dengan cara ini, Anda dapat



mengubah permintaan dalam penerbangan, proses permintaan debug, pengecualian tampilan, dan banyak lagi.

Contoh berikut menunjukkan pencegat sederhana yang menambahkan header tambahan ke semua permintaan keluar sebelum loop coba lagi dimasukkan.

```
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()
    }
}
```

Untuk informasi selengkapnya dan kait intersepsi yang tersedia, lihat [Interceptor](#).

## Pendaftaran pencegat

Anda mendaftarkan pencegat ketika Anda membangun klien layanan atau ketika Anda mengganti konfigurasi untuk serangkaian operasi tertentu.

Interceptor untuk semua operasi klien layanan

Kode berikut menambahkan `AddHeader` instance ke properti pencegat pembangun. Penambahan ini menambahkan `x-foo-version` header ke semua operasi sebelum loop coba lagi dimasukkan.

```
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 { ... }
```

## Pencegat hanya untuk operasi tertentu

Dengan menggunakan `withConfig` ekstensi, Anda dapat [mengganti konfigurasi klien layanan](#) untuk satu atau beberapa operasi untuk klien layanan apa pun. Dengan kemampuan ini, Anda dapat mendaftarkan pencegat tambahan untuk subset operasi.

Contoh berikut mengesampingkan konfigurasi s3 instance untuk operasi dalam ekstensi. `use` Operasi yang dipanggil `s3Scoped` berisi header `x-foo-version` dan `x-bar-version` header.

```
// '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.
}
```

## Menerapkan versi TLS minimum

Dengan AWS SDK for Kotlin, Anda dapat mengonfigurasi versi TLS minimum saat Anda terhubung ke titik akhir layanan. SDK menawarkan opsi konfigurasi yang berbeda. Dalam urutan prioritas tertinggi hingga terendah, pilihannya adalah:

- Konfigurasi mesin HTTP secara eksplisit
- Mengatur properti `sdk.minTls` sistem JVM
- Mengatur variabel `SDK_MIN_TLS` lingkungan

## Konfigurasi mesin HTTP

Ketika Anda menentukan mesin HTTP non-default untuk klien layanan, Anda dapat mengatur `tlsContext.minVersion` bidang.

Contoh berikut mengkonfigurasi mesin HTTP dan klien layanan apa pun yang menggunakannya untuk menggunakan TLS v1.2 minimal.

```
DynamoDbClient {
    region = "us-east-2"
```

```
httpClient {
    tlsContext {
        minVersion = TlsVersion.TLS_1_2
    }
}
}.use { ddb ->

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

## Mengatur properti **sdk.minTls** sistem JVM

Anda dapat mengatur properti sistem `sdk.minTls` JVM. Saat Anda meluncurkan aplikasi dengan set properti sistem, semua mesin HTTP dibangun dengan AWS SDK for Kotlin menggunakan versi TLS minimum yang ditentukan secara default. Namun, Anda dapat secara eksplisit mengganti ini dalam konfigurasi mesin HTTP. Nilai yang diijinkan adalah:

- TLS\_1\_0
- TLS\_1\_1
- TLS\_1\_2
- TLS\_1\_3

## Mengatur variabel **SDK\_MIN\_TLS** lingkungan

Anda dapat mengatur variabel `SDK_MIN_TLS` lingkungan. Saat Anda meluncurkan aplikasi dengan set variabel lingkungan, semua mesin HTTP dibangun dengan AWS SDK for Kotlin menggunakan versi TLS minimum yang ditentukan, kecuali diganti dengan opsi lain.

Nilai yang diijinkan adalah:

- TLS\_1\_0
- TLS\_1\_1
- TLS\_1\_2
- TLS\_1\_3

## Percobaan ulang

Panggilan untuk Layanan AWS sesekali mengembalikan pengecualian yang tidak terduga. Jenis kesalahan tertentu, seperti kesalahan pelambatan atau transien, mungkin berhasil jika panggilan dicoba ulang.

Halaman ini menjelaskan cara mengonfigurasi percobaan ulang otomatis dengan file. AWS SDK for Kotlin

### Konfigurasi coba lagi default

Secara default, setiap klien layanan secara otomatis dikonfigurasi dengan [strategi coba ulang standar](#). Konfigurasi default mencoba panggilan yang gagal hingga tiga kali (upaya awal ditambah dua percobaan ulang). Penundaan intervensi antara setiap panggilan dikonfigurasi dengan backoff eksponensial dan jitter acak untuk menghindari badai coba lagi. Konfigurasi ini berfungsi untuk sebagian besar kasus penggunaan tetapi mungkin tidak cocok dalam beberapa keadaan, seperti sistem throughput tinggi.

SDK mencoba mencoba ulang hanya pada kesalahan yang dapat dicoba ulang. Contoh kesalahan yang dapat dicoba ulang adalah batas waktu soket, pelambatan sisi layanan, kegagalan kunci konkurensi atau optimis, dan kesalahan layanan sementara. Parameter yang hilang atau tidak valid, kesalahan otentikasi/keamanan, dan pengecualian salah konfigurasi tidak dianggap dapat dicoba ulang.

Anda dapat menyesuaikan strategi coba ulang standar dengan menyetel upaya maksimum, penundaan dan backoff, dan konfigurasi bucket token.

### Upaya maksimal

Anda dapat menyesuaikan upaya maksimum default (3) di [blok retryStrategy DSL](#) selama konstruksi klien.

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

Dengan klien layanan DynamoDB yang ditampilkan dalam cuplikan sebelumnya, SDK mencoba panggilan API yang gagal hingga lima kali (upaya awal ditambah empat percobaan ulang).

Anda dapat menonaktifkan percobaan ulang otomatis sepenuhnya dengan mengatur upaya maksimum ke satu seperti yang ditunjukkan pada cuplikan berikut.

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

## Penundaan dan backoff

Jika percobaan ulang diperlukan, strategi coba lagi default menunggu sebelum melakukan upaya berikutnya. Penundaan untuk percobaan ulang pertama kecil tetapi tumbuh secara eksponensial untuk percobaan ulang nanti. Jumlah penundaan maksimum dibatasi sehingga tidak tumbuh terlalu besar.

Akhirnya, jitter acak diterapkan pada penundaan di antara semua upaya. Jitter membantu mengurangi efek armada besar yang dapat menyebabkan badai coba lagi. (Lihat [posting Blog AWS Arsitektur](#) ini untuk diskusi yang lebih dalam tentang backoff dan jitter eksponensial.)


Parameter penundaan dapat dikonfigurasi di blok [delayProviderDSL](#).

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

Dengan konfigurasi yang ditunjukkan pada cuplikan sebelumnya, klien menunda upaya percobaan ulang pertama hingga 100 milidetik. Jumlah waktu maksimum antara setiap upaya coba lagi adalah 5 detik.

Parameter berikut tersedia untuk tuning delay dan backoff.

Parameter	Nilai default	Deskripsi
<code>initialDelay</code>	10 milidetik	Jumlah maksimum keterlambatan untuk percobaan ulang

Parameter	Nilai default	Deskripsi
		pertama. Ketika jitter diterapkan, jumlah penundaan yang sebenarnya mungkin lebih sedikit.
<code> jitter </code>	1.0 (jitter penuh)	<p>Amplitudo maksimum yang digunakan untuk secara acak mengurangi penundaan yang dihitung. Nilai default 1.0 berarti bahwa penundaan yang dihitung dapat dikurangi menjadi jumlah berapa pun hingga 100% (misalnya, turun ke 0). Nilai 0,5 berarti bahwa penundaan yang dihitung dapat dikurangi hingga setengahnya. Dengan demikian, penundaan maksimal 10 ms dapat dikurangi menjadi antara 5 ms dan 10 ms. Nilai 0,0 berarti tidak ada jitter yang diterapkan.</p> <div data-bbox="1068 1325 1507 1686" style="border: 1px solid #f08080; border-radius: 10px; padding: 10px;"><p> <b>Important</b></p><p># Konfigurasi Jitter adalah fitur lanjutan. Menyesuaikan perilaku ini biasanya tidak disarankan.</p></div>

Parameter	Nilai default	Deskripsi
maxBackoff	20 detik	Jumlah maksimum keterlambatan untuk diterapkan pada upaya apa pun. Menetapkan nilai ini membatasi pertumbuhan eksponensial yang terjadi antara upaya berikutnya dan mencegah maksimum yang dihitung menjadi terlalu besar. Parameter ini membatasi penundaan yang dihitung sebelum jitter diterapkan. Jika diterapkan, jitter dapat mengurangi penundaan lebih jauh.

Parameter	Nilai default	Deskripsi
<code>scaleFactor</code>	1.5	<p>Basis eksponensial dimana penundaan maksimum berikutnya akan ditingkatkan. Misalnya, diberikan 10 ms dan 1,5, <code>scaleFactor</code> penundaan maksimal berikut akan dihitung: <code>initialDelay</code></p> <ul style="list-style-type: none"><li>• Coba lagi 1: <math>10\text{ms} \times 1.5^0 = 10\text{ms}</math></li><li>• Coba lagi 2: <math>10\text{ms} \times 1.5^1 = 15\text{ms}</math></li><li>• Coba lagi 3: <math>10\text{ms} \times 1.5^2 = 22,5\text{ ms}</math></li><li>• Coba lagi 4: <math>10\text{ms} \times 1.5^3 = 33.75\text{ms}</math></li></ul> <p>Ketika jitter diterapkan, jumlah sebenarnya dari setiap penundaan mungkin lebih sedikit.</p>

## Coba lagi ember token

Anda dapat memodifikasi perilaku strategi coba ulang standar lebih lanjut dengan menyesuaikan konfigurasi bucket token default. Bucket token coba lagi membantu mengurangi percobaan ulang yang cenderung tidak berhasil atau yang mungkin membutuhkan lebih banyak waktu untuk diselesaikan, seperti kegagalan batas waktu dan pembatasan.



**⚠ Important**

Konfigurasi token bucket adalah fitur lanjutan. Menyesuaikan perilaku ini biasanya tidak disarankan.

Setiap upaya coba lagi (opsional termasuk upaya awal) mengurangi beberapa kapasitas dari ember token. Jumlah yang dikurangi tergantung pada jenis upaya. Misalnya, mencoba ulang kesalahan sementara mungkin murah, tetapi mencoba kembali kesalahan batas waktu atau pelambatan mungkin lebih mahal.

Upaya yang berhasil mengembalikan kapasitas ke ember. Bucket mungkin tidak bertambah melebihi kapasitas maksimumnya atau dikurangi di bawah nol.

Bergantung pada nilai `useCircuitBreakerMode` pengaturan, upaya untuk mengurangi kapasitas di bawah nol menghasilkan salah satu hasil berikut:

- Jika pengaturannya BENAR, pengecualian dilemparkan— Misalnya, jika terlalu banyak percobaan ulang telah terjadi dan lebih banyak percobaan ulang tidak mungkin berhasil.
- Jika pengaturannya SALAH, ada penundaan — Misalnya, penundaan hingga bucket memiliki kapasitas yang cukup lagi.

Parameter bucket token dapat dikonfigurasi di blok [tokenBucketDSL](#):

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

Parameter berikut tersedia untuk menyetel ember token coba lagi:

Parameter	Nilai default	Deskripsi
<code>initialTryCost</code>	0	Jumlah yang akan dikurangi dari ember untuk upaya awal.

Parameter	Nilai default	Deskripsi
		Nilai default 0 berarti tidak ada kapasitas yang akan dikurangi dan dengan demikian upaya awal tidak dihentikan atau ditunda.
<code>initialTrySuccessIncrement</code>	1	Jumlah untuk meningkatkan kapasitas ketika upaya awal berhasil.
<code>maxCapacity</code>	500	Kapasitas maksimum ember token. Jumlah token yang tersedia tidak dapat melebihi jumlah ini.
<code>refillUnitsPerSecond</code>	0	Jumlah kapasitas ditambahkan kembali ke ember setiap detik. Nilai 0 berarti tidak ada kapasitas yang ditambahkan kembali secara otomatis. (Misalnya, hanya upaya yang berhasil yang menghasilkan peningkatan kapasitas). Nilai 0 <code>useCircuitBreakerMode</code> harus BENAR.
<code>retryCost</code>	5	Jumlah yang akan dikurangi dari ember untuk upaya setelah kegagalan sementara. Jumlah yang sama ditambahkan kembali ke ember jika upaya berhasil.

Parameter	Nilai default	Deskripsi
<code>timeoutRetryCost</code>	10	Jumlah yang akan dikurangi dari bucket untuk upaya setelah batas waktu atau kegagalan pelambatan. Jumlah yang sama ditambahkan kembali ke ember jika upaya berhasil.
<code>useCircuitBreakerMode</code>	BETUL	Menentukan perilaku ketika upaya untuk mengurangi kapasitas akan mengakibatkan kapasitas bucket turun di bawah nol. Ketika TRUE, bucket token akan mengeluarkan pengecualian yang menunjukkan bahwa tidak ada lagi kapasitas coba lagi. Ketika FALSE, token bucket akan menunda upaya sampai kapasitas yang cukup telah diisi ulang.

## Coba ulang adaptif

Sebagai alternatif dari strategi coba ulang standar, strategi coba lagi adaptif adalah pendekatan lanjutan yang mencari tingkat permintaan yang ideal untuk meminimalkan kesalahan pelambatan.

### Important

Coba ulang adaptif adalah mode coba lagi lanjutan. Menggunakan strategi coba lagi ini biasanya tidak disarankan.

Percobaan ulang adaptif mencakup semua fitur percobaan ulang standar. Ini menambahkan pembatas tingkat sisi klien yang mengukur tingkat permintaan yang dibatasi dibandingkan dengan

permintaan non-throttled. Ini juga membatasi lalu lintas untuk mencoba tetap berada dalam bandwidth yang aman, idealnya menyebabkan kesalahan pelambatan nol.

Tarif beradaptasi secara real time untuk mengubah kondisi layanan dan pola lalu lintas dan dapat meningkatkan atau menurunkan tingkat lalu lintas yang sesuai. Secara kritis, pembatas tarif mungkin menunda upaya awal dalam skenario lalu lintas tinggi.

Anda memilih strategi coba ulang adaptif dengan memberikan parameter tambahan pada `retryStrategy` metode ini. Parameter pembatas laju dapat dikonfigurasi di blok [rateLimiterDSL](#).

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

#### Note

Strategi coba ulang adaptif mengasumsikan bahwa klien bekerja melawan satu sumber daya (misalnya, satu tabel DynamoDB atau satu bucket Amazon S3).

Jika Anda menggunakan satu klien untuk beberapa sumber daya, pembatasan atau pemadaman yang terkait dengan satu sumber daya menghasilkan peningkatan latensi dan kegagalan saat klien mengakses semua sumber daya lainnya. Saat Anda menggunakan strategi coba ulang adaptif, kami sarankan Anda menggunakan satu klien untuk setiap sumber daya.

## Observabilitas

Observabilitas adalah sejauh mana keadaan sistem saat ini dapat disimpulkan dari data yang dipancarkannya. Data yang dipancarkan biasanya disebut sebagai telemetri.

AWS SDK for Kotlin Dapat menyediakan ketiga sinyal telemetri umum: metrik, jejak, dan log. Anda dapat menghubungkan data telemetri [TelemetryProvider](#) untuk mengirim ke backend observabilitas (seperti atau [AWS X-Ray/Amazon CloudWatch](#)) dan kemudian menindaklanjutinya.

Secara default, hanya logging yang diaktifkan dan sinyal telemetry lainnya dinonaktifkan di SDK. Topik ini menjelaskan cara mengaktifkan dan mengonfigurasi keluaran telemetry.

### Important

`TelemetryProvider` saat ini merupakan API eksperimental yang harus dipilih untuk digunakan.

## Konfigurasi a `TelemetryProvider`

Anda dapat mengonfigurasi a `TelemetryProvider` dalam aplikasi Anda secara global untuk semua klien layanan atau untuk klien individu. Contoh berikut menggunakan `getConfiguredProvider()` fungsi hipotetis untuk mendemonstrasikan operasi `TelemetryProvider` API. [the section called “Penyedia telemetry”](#) Bagian ini menjelaskan informasi untuk implementasi yang disediakan oleh SDK. Jika penyedia tidak didukung, Anda dapat menerapkan dukungan Anda sendiri atau [membuka permintaan fitur pada GitHub](#).

### Konfigurasi penyedia telemetry global default

Secara default, setiap klien layanan mencoba menggunakan penyedia telemetry yang tersedia secara global. Dengan cara ini, Anda dapat mengatur penyedia sekali, dan semua klien akan menggunakannya. Ini harus dilakukan hanya sekali, sebelum Anda membuat instance klien layanan apa pun.

Untuk menggunakan penyedia telemetry global, perbarui dependensi project terlebih dahulu untuk menambahkan modul default telemetry seperti yang ditunjukkan pada cuplikan Gradle berikut.

(Anda dapat menavigasi ke [X.Y.Z](#) tautan untuk melihat versi terbaru yang tersedia.)

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

Kemudian atur penyedia telemetry global sebelum membuat klien layanan seperti yang ditunjukkan pada kode berikut.

```
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")
}
```

## Konfigurasi penyedia telemetri untuk klien layanan tertentu

Anda dapat mengonfigurasi klien layanan individual dengan penyedia telemetri tertentu (selain yang global). Seperti yang ditunjukkan dalam contoh berikut.

```
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")
}
```

## Metrik

Tabel berikut mencantumkan metrik telemetri yang dipancarkan SDK. [Konfigurasi penyedia telemetri](#) untuk membuat metrik dapat diamati.

## Metrik apa yang dipancarkan?

Nama metrik	Unit	Tipe	Atribut	Deskripsi
<code>smithy.client.call.duration</code>	detik	Histogram	<code>rpc.service</code> , <code>rpc.method</code>	Durasi panggilan keseluruhan (termasuk percobaan ulang)
<code>smithy.client.call.attempts</code>	{mencoba}	Monoton Counter	<code>rpc.service</code> , <code>rpc.method</code>	Jumlah upaya untuk operasi individu
<code>smithy.client.call.errors</code>	{kesalahan}	Monoton Counter	<code>rpc.service</code> , <code>rpc.method</code> , <code>exception.type</code>	Jumlah kesalahan untuk suatu operasi
<code>smithy.client.call.attempt_duration</code>	detik	Histogram	<code>rpc.service</code> , <code>rpc.method</code>	Waktu yang diperlukan untuk terhubung ke layanan, mengirim permintaan, dan mendapatkan kembali kode status HTTP dan header (termasuk waktu antrian menunggu untuk dikirim)
<code>smithy.client.call.resolve_endpoint_duration</code>	detik	Histogram	<code>rpc.service</code> , <code>rpc.method</code>	Waktu yang dibutuhkan untuk menyelesaikan titik akhir (endpoint resolver, bukan DNS) untuk permintaan
<code>smithy.client.call.serialization_duration</code>	detik	Histogram	<code>rpc.service</code> , <code>rpc.method</code>	Waktu yang dibutuhkan untuk membuat serial badan pesan
<code>smithy.client.call.deserialization_duration</code>	detik	Histogram	<code>rpc.service</code> , <code>rpc.method</code>	Waktu yang dibutuhkan untuk deserialisasi badan pesan
<code>smithy.client.call.auth.signing_duration</code>	detik	Histogram	<code>rpc.service</code> , <code>rpc.method</code> , <code>auth.scheme_id</code>	Waktu yang dibutuhkan untuk menandatangani permintaan

Nama metrik	Unit	Tipe	Atribut	Deskripsi
smithy.client.call .auth.resolution_duration	detik	Histogram	rpc.service, rpc.method, auth.scheme_id	Waktu yang dibutuhkan untuk memperoleh identitas (seperti AWS kredensial atau token pembawa) dari Penyedia Identitas
smithy.client.http .connections_acquisition_duration	detik	Histogram		Waktu yang dibutuhkan permintaan untuk mendapatkan koneksi
smithy.client.http .connections_limit	{koneksi}	[Asinkron] UpDownCounter		Koneksi terbuka maksimum yang diizinkan/dikonfigurasi untuk klien HTTP
smithy.client.http .connections_usage	{koneksi}	[Asinkron] UpDownCounter	negara: idle   diperoleh	Status koneksi saat ini
smithy.client.http .connections_uptime	detik	Histogram		Jumlah waktu koneksi telah terbuka
smithy.client.http .requests_usage	{permintaan}	[Asinkron] UpDownCounter	negara bagian: antri   dalam penerbangan	Status saat ini dari permintaan klien HTTP konkurensi
smithy.client.http .requests_queued_duration	detik	Histogram		Jumlah waktu permintaan yang dihabiskan antrian dan menunggu untuk dieksekusi oleh klien HTTP
smithy.client.http .bytes_sent	Oleh	Monoton Counter	server.address	Jumlah total byte yang dikirim oleh klien HTTP



Nama metrik	Unit	Tipe	Atribut	Deskripsi
smithy.client.http.bytes_received	Oleh	Monoton Counter	server.address	Jumlah total byte yang diterima oleh klien HTTP

Berikut ini adalah deskripsi kolom:

- Nama metrik —Nama metrik yang dipancarkan.
- Satuan —Satuan ukuran untuk metrik. Unit diberikan dalam notasi [UCUM](#) case sensitive (“c/s”).
- Jenis —Jenis instrumen yang digunakan untuk menangkap metrik.
- Deskripsi —Deskripsi tentang apa yang diukur metrik.
- Atribut —Himpunan atribut (dimensi) yang dipancarkan dengan metrik.

## Pencatatan log

AWS SDK for Kotlin Mengkonfigurasi logger yang kompatibel dengan [SLF4J](#) sebagai default `LoggerProvider` penyedia telemetri. Dengan SLF4 J, yang merupakan lapisan abstraksi, Anda dapat menggunakan salah satu dari beberapa sistem logging saat runtime. [Sistem logging yang didukung termasuk Java Logging APIs, Log4j 2, dan Logback.](#)

### Warning

Kami menyarankan Anda hanya menggunakan wire logging untuk tujuan debugging. (Wire logging dibahas di bawah.) Matikan di lingkungan produksi Anda karena dapat mencatat data sensitif seperti alamat email, token keamanan, kunci API, kata sandi, dan AWS Secrets Manager rahasia. Wire logging mencatat permintaan atau respons penuh tanpa enkripsi, bahkan untuk panggilan HTTPS.

Untuk permintaan besar (seperti mengunggah file ke Amazon S3) atau tanggapan, pencatatan kawat verbose juga dapat memengaruhi kinerja aplikasi Anda secara signifikan.

### Contoh konfigurasi logging Log4j 2

Meskipun pustaka log SLF4J yang kompatibel dapat digunakan, contoh ini memungkinkan keluaran log dari SDK dalam program JVM menggunakan Log4j 2:

### Ketergantungan gradle

(Anda dapat menavigasi ke [X.Y.Z](#) tautan untuk melihat versi terbaru yang tersedia.)

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

## File konfigurasi Log4j 2

Buat file bernama `log4j2.xml` di `resources` direktori Anda (misalnya, `<project-dir>/src/main/resources`). Tambahkan konfigurasi XHTML berikut ke file:

```
<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>
```

Konfigurasi ini mencakup `%X` penentu dalam `pattern` atribut yang memungkinkan pencatatan MDC (konteks diagnostik yang dipetakan).

SDK menambahkan elemen MDC berikut untuk setiap operasi.

`rpc`

Nama RPC yang dipanggil, misalnya, `S3.GetObject`

`sdkInvocationId`

ID unik yang ditetapkan oleh klien layanan untuk operasi. ID mengkorelasikan semua peristiwa logging yang terkait dengan pemanggilan satu operasi.

## Tentukan mode log untuk pesan tingkat kabel

Secara default, AWS SDK for Kotlin tidak mencatat pesan tingkat kabel karena mungkin berisi data sensitif dari permintaan dan tanggapan API. Namun, terkadang Anda memerlukan tingkat detail ini untuk tujuan debugging.

Dengan Kotlin SDK, Anda dapat mengatur mode log dalam kode atau menggunakan pengaturan lingkungan untuk mengaktifkan pesan debug untuk hal-hal berikut:

- Permintaan HTTP
- Tanggapan HTTP

Mode log didukung oleh bit-field di mana setiap bit adalah flag (mode) dan nilainya aditif. Anda dapat menggabungkan satu mode permintaan dan satu mode respons.

Atur mode log dalam kode

Untuk memilih logging tambahan, setel `LogMode` properti saat Anda membangun klien layanan.

Contoh berikut menunjukkan cara mengaktifkan pencatatan permintaan (dengan badan) dan respons (tanpa badan).

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

// ...

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

Nilai mode log yang ditetapkan selama konstruksi klien layanan, mengesampingkan nilai mode log apa pun yang ditetapkan dari lingkungan.

Atur mode log dari lingkungan

Untuk mengatur mode log secara global untuk semua klien layanan yang tidak dikonfigurasi secara eksplisit dalam kode, gunakan salah satu dari berikut ini:

- Properti sistem JVM: `sdk.logMode`
- Variabel lingkungan: `SDK_LOG_MODE`

Nilai case-insensitive berikut tersedia:

- `LogRequest`
- `LogRequestWithBody`

- `LogResponse`
- `LogResponseWithBody`

Untuk membuat mode log gabungan menggunakan pengaturan dari lingkungan, Anda memisahkan nilai dengan simbol pipe (`|`).

Misalnya, contoh berikut mengatur mode log yang sama dengan contoh sebelumnya.

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

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

#### Note

Anda juga harus mengonfigurasi logger SLF4 J yang kompatibel dan mengatur level logging ke `DEBUG` untuk mengaktifkan logging tingkat kabel.

## Penyedia telemetri

SDK saat ini mendukung [OpenTelemetry](#) (OTel) sebagai penyedia. SDK mungkin menawarkan penyedia telemetri tambahan di masa depan.

Topik

- [Konfigurasi penyedia telemetri OpenTelemetry berbasis](#)

## Konfigurasi penyedia telemetri OpenTelemetry berbasis

SDK untuk Kotlin menyediakan implementasi `TelemetryProvider` antarmuka yang didukung oleh `OpenTelemetry`

Prasyarat

Perbarui dependensi project Anda untuk menambahkan `OpenTelemetry` penyedia seperti yang ditunjukkan pada cuplikan Gradle berikut. Anda dapat menavigasi ke [X](#), [Y](#), [Z](#) tautan untuk melihat versi terbaru yang tersedia.

```
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")
}
```

## Konfigurasi SDK

Kode berikut mengkonfigurasi klien layanan dengan menggunakan penyedia OpenTelemetry telemetry.

```
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

Diskusi tentang cara mengkonfigurasi OpenTelemetry SDK berada di luar cakupan panduan ini. [Dokumentasi OpenTelemetry Java](#) berisi informasi konfigurasi pada berbagai pendekatan: [secara manual](#), secara otomatis melalui [agen Java](#), atau [kolektor](#) (opsional).

## Sumber daya

Sumber daya berikut tersedia untuk membantu Anda memulai OpenTelemetry.

- [AWS Distro untuk OpenTelemetry](#) - beranda AWS OTe L Distro
- [aws-otel-java-instrumentation](#)- AWS Distro untuk Perpustakaan Instrumentasi OpenTelemetry Java
- [aws-otel-lambda](#)- Lapisan OpenTelemetry Lambda yang AWS dikelola
- [aws-otel-collector](#)- AWS Distro untuk Kolektor OpenTelemetry
- [AWS Praktik Terbaik Observabilitas - Praktik](#) terbaik umum untuk observabilitas khusus untuk AWS

## Ganti konfigurasi klien layanan

Setelah [klien layanan dibuat](#), klien layanan menggunakan konfigurasi tetap untuk semua operasi. Namun, terkadang Anda mungkin perlu mengganti konfigurasi untuk satu atau lebih operasi spesifik.

Setiap klien layanan memiliki `withConfig` ekstensi sehingga Anda dapat memodifikasi salinan konfigurasi yang ada. `withConfigEkstensi` mengembalikan klien layanan baru dengan konfigurasi yang dimodifikasi. Klien asli ada secara independen dan menggunakan konfigurasi aslinya.

Contoh berikut menunjukkan pembuatan `S3Client` instance yang memanggil dua operasi.

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

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

Cuplikan berikut menunjukkan cara mengganti konfigurasi untuk satu operasi. `listObjectV2`

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

Panggilan operasi pada `s3` klien menggunakan konfigurasi asli yang ditentukan saat klien dibuat. Konfigurasinya mencakup [pencatatan permintaan](#) dan `us-west-2` region untuk Wilayah.

`listObjectsV2` Pemanggilan pada `overriddenS3` klien menggunakan pengaturan yang sama dengan `s3` klien asli kecuali untuk Wilayah, yang sekarang. `eu-central-1`

## Siklus hidup klien yang diganti

Dalam contoh sebelumnya, `s3` klien dan `overriddenS3` klien independen satu sama lain. Operasi dapat dipanggil pada salah satu klien selama mereka tetap terbuka. Masing-masing menggunakan konfigurasi terpisah, tetapi mereka dapat berbagi sumber daya yang mendasarinya (seperti mesin HTTP) kecuali jika itu juga diganti.

Anda menutup klien dengan konfigurasi yang diganti dan klien asli secara terpisah. Anda dapat menutup klien dengan konfigurasi yang diganti sebelum atau setelah Anda menutup klien aslinya. Kecuali Anda perlu menggunakan klien dengan konfigurasi yang diganti untuk waktu yang lama, kami sarankan Anda membungkus siklus hidupnya dengan metode ini. `use useMetode` ini memastikan bahwa klien ditutup jika pengecualian terjadi.

## Sumber daya yang dibagikan antar klien

Ketika Anda membuat klien layanan dengan menggunakan `withConfig`, itu mungkin berbagi sumber daya dengan klien asli. Sebaliknya, ketika Anda membuat klien dengan menggunakan [FromEnvironment](#) atau Anda [secara eksplisit mengonfigurasinya](#), klien menggunakan sumber daya independen. Sumber daya seperti mesin HTTP dan penyedia kredensial dibagikan kecuali jika diganti di blok. `withConfig`

Karena siklus hidup setiap klien independen, sumber daya bersama tetap terbuka dan dapat digunakan sampai klien terakhir ditutup. Oleh karena itu, penting bagi Anda untuk menutup klien layanan yang diganti ketika Anda tidak lagi membutuhkannya. Ini mencegah sumber daya bersama tetap terbuka dan mengkonsumsi sumber daya sistem seperti memori, koneksi, dan siklus CPU.

Contoh berikut menunjukkan sumber daya bersama dan independen.

`overriddenS3Klien s3` dan berbagi instance penyedia kredensial yang sama, termasuk konfigurasi caching. Panggilan yang dilakukan dengan `overriddenS3` menggunakan kembali kredensial jika nilai cache masih terkini dari panggilan yang dilakukan oleh klien. `s3`

Mesin HTTP tidak dibagi antara dua klien. Setiap klien memiliki mesin HTTP independen karena diganti dalam panggilan. `withConfig`

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

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



# Gunakan SDK

Bagian ini memberikan informasi dasar yang diperlukan untuk menggunakan AWS SDK for Kotlin.

Topik

- [Membuat permintaan](#)
- [Coroutine](#)
- [Operasi streaming](#)
- [Paginasi](#)
- [Pelayan](#)
- [Penanganan kesalahan](#)
- [Permintaan presign](#)
- [Pemecahan Masalah FAQs](#)

## Membuat permintaan

Gunakan klien layanan untuk membuat permintaan ke Layanan AWS. AWS SDK for Kotlin Menyediakan Domain Specific Languages (DSLs) mengikuti pola [pembuat type-safe](#) untuk membuat permintaan. Struktur permintaan bersarang juga dapat diakses melalui permintaan merekaDSLs.

Contoh berikut menunjukkan cara membuat input operasi Amazon [createTable](#)DynamoDB:

```
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)

```

## DSLKelebihan antarmuka layanan

Setiap operasi non-streaming pada antarmuka klien layanan memiliki DSL kelebihan sehingga Anda tidak perlu membuat permintaan terpisah.

Contoh pembuatan bucket Amazon Simple Storage Service (Amazon S3) dengan fungsi overloaded:

```

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

```

Ini setara dengan:

```

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

s3client.createBucket(request)

```

## Permintaan tanpa input yang diperlukan

Operasi yang tidak memiliki input yang diperlukan dapat dipanggil tanpa harus melewati objek permintaan. Ini sering dimungkinkan dengan operasi tipe daftar, seperti operasi Amazon `listBuckets` API S3.

Misalnya, tiga pernyataan berikut ini setara:

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

## Coroutine

AWS SDK for Kotlin ini asinkron secara default. The SDK for Kotlin menggunakan `suspend` fungsi untuk semua operasi, yang dimaksudkan untuk dipanggil dari coroutine.

Untuk panduan lebih mendalam tentang coroutine, lihat dokumentasi [resmi](#) Kotlin.

## Membuat permintaan bersamaan

Pembuat coroutine [async](#) dapat digunakan untuk meluncurkan permintaan bersamaan di mana Anda peduli dengan hasilnya. `async` mengembalikan [Deferred](#), yang mewakili masa depan yang ringan dan tidak memblokir yang mewakili janji untuk memberikan hasil nanti.

Jika Anda tidak peduli dengan hasilnya (hanya bahwa operasi selesai), Anda dapat menggunakan pembuat coroutine [peluncuran](#). `launch` secara konseptual mirip dengan `async`. Perbedaannya adalah peluncuran mengembalikan [Job](#) dan tidak membawa nilai apa pun yang dihasilkan, sementara `async` mengembalikan `Deferred`.

Berikut ini adalah contoh membuat permintaan bersamaan ke Amazon S3 menggunakan [headObject](#) operasi untuk mendapatkan ukuran konten dua kunci:

```
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")
}
```

## Membuat permintaan pemblokiran

Untuk melakukan panggilan layanan dari kode yang ada yang tidak menggunakan coroutine dan mengimplementasikan model threading yang berbeda, Anda dapat menggunakan pembuat coroutine. [runBlocking](#) Contoh model threading yang berbeda menggunakan pendekatan eksekutor/futures

tradisional Java. Anda mungkin perlu menggunakan pendekatan ini jika Anda memadukan kode atau pustaka Java dan Kotlin.

Seperti namanya, `runBlocking` pembuat ini meluncurkan coroutine baru dan memblokir utas saat ini hingga selesai.

#### Warning

`runBlocking` umumnya tidak boleh digunakan dari coroutine. Ini dirancang untuk menjembatani kode pemblokiran reguler ke perpustakaan yang ditulis dalam gaya penangguhan (seperti dalam fungsi utama dan tes).

## Operasi streaming

Dalam AWS SDK for Kotlin, data biner (aliran) direpresentasikan sebagai [ByteStream](#) tipe, yang merupakan aliran byte hanya-baca abstrak.

## Respons streaming

Respons dengan aliran biner (seperti operasi Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon [GetObject](#) API S3)) ditangani secara berbeda dari metode lain. Metode ini mengambil fungsi lambda yang menangani respons daripada mengembalikan respons secara langsung. Ini membatasi ruang lingkup respons terhadap fungsi dan menyederhanakan manajemen seumur hidup untuk penelepon dan runtime. SDK

Setelah fungsi lambda kembali, sumber daya apa pun seperti HTTP koneksi yang mendasarinya dilepaskan. (`ByteStream` seharusnya tidak diakses setelah lambda kembali dan tidak boleh dilewatkan dari penutupan.) Hasil panggilan adalah apa pun yang dikembalikan lambda.

Contoh kode berikut menunjukkan [getObject](#) fungsi yang menerima parameter lambda, yang menangani respons.

```
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 contentType = 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 $contentType bytes to $path")
```

ByteStreamJenis ini memiliki ekstensi berikut untuk cara umum mengkonsumsinya:

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

Semua ini didefinisikan dalam `aws.smithy.kotlin.runtime.content` paket.

## Permintaan streaming

Untuk memasok aByteStream, ada juga beberapa metode kenyamanan, termasuk yang berikut:

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

Semua ini didefinisikan dalam `aws.smithy.kotlin.runtime.content` paket.

Contoh kode berikut menunjukkan penggunaan metode ByteStream kenyamanan yang menyediakan properti tubuh dalam pembuatan [PutObjectRequest](#):

```
val req = PutObjectRequest {
```

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

## Paginasi

Banyak AWS operasi mengembalikan hasil paginasi ketika muatan terlalu besar untuk dikembalikan dalam satu respons. AWS SDK for Kotlin Termasuk [ekstensi](#) ke antarmuka klien layanan yang secara otomatis melakukan paginasi hasil untuk Anda. Anda hanya perlu menulis kode yang memproses hasilnya.

Pagination diekspos sebagai [Flow](#) <T> sehingga Anda dapat memanfaatkan transformasi idiomatik Kotlin untuk koleksi asinkron (seperti, dan). `map` `filter` `take` Pengecualian bersifat transparan, yang membuat penanganan kesalahan terasa seperti API panggilan biasa, dan pembatalan mematuhi pembatalan koperasi umum coroutine. Untuk informasi selengkapnya, lihat [pengecualian aliran dan aliran](#) di panduan resmi.

### Note

Contoh berikut menggunakan Amazon S3. Namun, konsepnya sama untuk layanan apa pun yang memiliki satu atau lebih paginasi APIs. Semua ekstensi pagination didefinisikan dalam `aws.sdk.kotlin.<service>.paginators` paket (seperti `aws.sdk.kotlin.dynamodb.paginators`).

Contoh kode berikut menunjukkan bagaimana Anda dapat memproses respons paginasi dari panggilan fungsi [listObjectsV2Paginated](#).

### Impor

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

### Kode

```
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}")
    }
```

## Pelayan

Pelayan adalah abstraksi sisi klien yang digunakan untuk polling sumber daya sampai keadaan yang diinginkan tercapai, atau sampai ditentukan bahwa sumber daya tidak akan memasuki keadaan yang diinginkan. Ini adalah tugas umum saat bekerja dengan layanan yang pada akhirnya konsisten, seperti Amazon Simple Storage Service (Amazon S3), atau layanan yang membuat sumber daya secara asinkron, seperti Amazon. EC2

Menulis logika untuk terus polling status sumber daya dapat menjadi rumit dan rawan kesalahan. Tujuan pelayan adalah untuk memindahkan tanggung jawab ini dari kode pelanggan dan ke dalam AWS SDK for Kotlin, yang memiliki pengetahuan mendalam tentang aspek waktu untuk operasi. AWS

### Note

Contoh berikut menggunakan Amazon S3. Namun, konsepnya sama untuk setiap Layanan AWS yang memiliki satu atau lebih pelayan didefinisikan. Semua ekstensi didefinisikan dalam `aws.sdk.kotlin.<service>.waiters` paket (seperti `aws.sdk.kotlin.dynamodb.waiters`). Mereka juga mengikuti konvensi penamaan standar (`waitUntil<Condition>`).

Contoh kode berikut menunjukkan penggunaan fungsi pelayan yang memungkinkan Anda menghindari penulisan logika polling.

### Impor



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

## Kode

```
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

Setiap metode tunggu mengembalikan Outcome instance yang dapat digunakan untuk mendapatkan respons akhir yang sesuai untuk mencapai kondisi yang diinginkan. Hasil juga berisi rincian tambahan seperti jumlah upaya yang dilakukan untuk mencapai keadaan yang diinginkan.

## Penanganan kesalahan

Memahami bagaimana dan kapan pengecualian AWS SDK for Kotlin melempar penting untuk membangun aplikasi berkualitas tinggi menggunakan SDK. Bagian berikut menjelaskan berbagai kasus pengecualian yang dilemparkan oleh SDK dan bagaimana menanganinya dengan tepat.

### Pengecualian layanan

Pengecualian yang paling umum adalah `AwsServiceException`, dari mana semua pengecualian khusus layanan (seperti `S3Exception`) mewarisi. Pengecualian ini merupakan respons kesalahan dari file Layanan AWS. Misalnya, jika Anda mencoba menghentikan EC2 instans Amazon yang tidak ada, Amazon EC2 mengembalikan respons kesalahan. Detail respons kesalahan disertakan dalam `AwsServiceException` yang dilemparkan.

Ketika Anda menemukan `AwsServiceException`, ini berarti bahwa permintaan Anda berhasil dikirim ke Layanan AWS tetapi tidak dapat diproses. Ini bisa karena kesalahan dalam parameter permintaan atau karena masalah di sisi layanan.

## Pengecualian klien

`ClientException` menunjukkan bahwa masalah terjadi di dalam kode AWS SDK for Kotlin klien, baik saat mencoba mengirim permintaan ke AWS atau saat mencoba mengurai respons dari AWS. A `ClientException` umumnya lebih parah daripada `AwsServiceException` dan menunjukkan bahwa masalah utama mencegah klien memproses panggilan layanan ke Layanan AWS. Misalnya, AWS SDK for Kotlin melempar a `ClientException` jika gagal mengurai respons dari layanan.

## Metadata kesalahan

Setiap pengecualian layanan dan pengecualian klien memiliki `sdkErrorMetadata` properti. Ini adalah tas properti yang diketik yang dapat digunakan untuk mengambil detail tambahan tentang kesalahan.

Beberapa ekstensi yang telah ditentukan ada untuk `AwsErrorMetadata` jenis secara langsung, termasuk namun tidak terbatas pada yang berikut:

- `sdkErrorMetadata.requestId`— id permintaan unik
- `sdkErrorMetadata.errorMessage`— pesan yang dapat dibaca manusia (biasanya cocok dengan `Exception.message`, tetapi mungkin berisi informasi lebih lanjut jika pengecualian tidak diketahui oleh layanan)
- `sdkErrorMetadata.protocolResponse`— Respons protokol mentah

Contoh berikut menunjukkan mengakses metadata kesalahan.

```
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}")
}
```

## Permintaan presign

Anda dapat melakukan presign permintaan untuk beberapa AWS API operasi sehingga pemanggil lain dapat menggunakan permintaan nanti tanpa menunjukkan kredensialnya sendiri.

Misalnya, asumsikan bahwa Alice memiliki akses ke objek Amazon Simple Storage Service (Amazon S3) dan dia ingin berbagi sementara akses objek dengan Bob. Alice dapat membuat `GetObject` permintaan presigned untuk berbagi dengan Bob sehingga ia dapat men-download objek tanpa memerlukan akses ke kredensi Alice.

## Dasar-dasar pra-penandatanganan

The SDK for Kotlin menyediakan metode ekstensi pada klien layanan untuk melakukan presign permintaan. Semua permintaan yang telah ditetapkan sebelumnya memerlukan durasi yang menunjukkan berapa lama permintaan yang ditandatangani valid. Setelah durasi berakhir, permintaan presigned berakhir dan memunculkan kesalahan otentikasi jika dijalankan.

Kode berikut menunjukkan contoh yang membuat `GetObject` permintaan yang telah ditetapkan sebelumnya untuk Amazon S3. Permintaan ini berlaku selama 24 jam setelah pembuatan.

```
val s3 = S3Client.fromEnvironment()

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

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

Metode [presignGetObject](#) ekstensi mengembalikan [HttpRequest](#) objek. Objek permintaan berisi presigned URL di mana operasi dapat dipanggil. Penelepon lain dapat menggunakan URL (atau seluruh permintaan) dalam basis kode atau lingkungan bahasa pemrograman yang berbeda.

Setelah membuat permintaan yang telah ditentukan sebelumnya, gunakan HTTP klien untuk memanggil permintaan. API untuk meminta HTTP GET permintaan tergantung pada HTTP klien. Contoh berikut menggunakan [URL.readText](#) metode Kotlin.

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

## Konfigurasi presigning lanjutan

Dalam SDK, setiap metode yang dapat melakukan presign permintaan memiliki kelebihan beban yang dapat Anda gunakan untuk menyediakan opsi konfigurasi lanjutan, seperti implementasi penandatanganan tertentu atau parameter penandatanganan terperinci.

Contoh berikut menunjukkan GetObject permintaan Amazon S3 yang menggunakan varian CRT penandatanganan dan menentukan tanggal penandatanganan future.

```
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
}
```

Permintaan presigned yang dikembalikan bertanggal 24 jam dan tidak valid sebelum itu. Kedaluwarsa 8 jam setelah itu.

## Presigning POST dan permintaan PUT

Banyak operasi yang dapat dipresignable hanya memerlukan a URL dan harus dieksekusi sebagai HTTP GET permintaan. Beberapa operasi, bagaimanapun, mengambil tubuh dan harus dieksekusi sebagai HTTP POST atau HTTP PUT permintaan bersama dengan header dalam beberapa kasus. Penandatanganan permintaan ini identik dengan permintaan GET presigning, tetapi memanggil permintaan yang telah ditetapkan sebelumnya lebih rumit.

Berikut adalah contoh penandatanganan permintaan S3PutObject:

```
val s3 = S3Client.fromEnvironment()

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

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

Yang dikembalikan `HttpRequest` memiliki nilai metode `HttpMethod.PUT` dan menyertakan a URL dan header yang harus disertakan dalam pemanggilan permintaan di masa depan. HTTP Anda dapat meneruskan permintaan ini ke penelepon yang dapat menjalankannya di basis kode atau lingkungan bahasa pemrograman yang berbeda.

Setelah membuat presigned POST atau PUT request, gunakan HTTP klien untuk memanggil permintaan. API untuk memanggil POST atau PUT meminta URL tergantung pada HTTP klien yang digunakan. Contoh berikut menggunakan [OkHttp HTTPklien](#) dan menyertakan badan yang berisi `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()
```

## Operasi yang SDK dapat dilakukan sebelumnya

SDK untuk Kotlin saat ini mendukung presigning API operasi berikut yang perlu dipanggil dengan metode terkait HTTP.

Layanan AWS	Operasi	SDK metode ekstensi	Gunakan HTTP metode
Amazon S3	GetObject	<a href="#">presignGetObject</a>	HTTP GET
Amazon S3	PutObject	<a href="#">presignPutObject</a>	HTTP PUT
Amazon S3	UploadPart	<a href="#">presignUploadPart</a>	HTTP PUT

Layanan AWS	Operasi	SDKmetode ekstensi	Gunakan HTTP metode
AWS Security Token Service	GetCallerIdentity	<a href="#">presignGetCallerId entitas</a>	HTTP POST
Amazon Polly	SynthesizeSpeech	<a href="#">presignSynthesizeSpeech</a>	HTTP POST

## Pemecahan Masalah FAQs

Saat Anda menggunakan aplikasi Anda, Anda mungkin mengalami beberapa masalah yang tercantum dalam topik ini. AWS SDK for Kotlin Gunakan saran berikut untuk membantu mengungkap akar penyebab dan mengatasi kesalahan.

### Bagaimana cara memperbaiki masalah “koneksi ditutup”?

Anda mungkin mengalami masalah “koneksi ditutup” sebagai pengecualian seperti salah satu jenis berikut:

- `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; HttpStatusCode(CONNECTION_CLOSED)`

Pengecualian ini menunjukkan bahwa TCP koneksi dari layanan SDK ke layanan tiba-tiba ditutup atau diatur ulang. Koneksi mungkin telah ditutup oleh host Anda, AWS layanan, atau pihak perantara seperti NAT gateway, proxy, atau penyeimbang beban.

Jenis pengecualian ini secara otomatis dicoba ulang tetapi mungkin masih muncul di SDK log, tergantung pada konfigurasi logging Anda. Jika pengecualian dilemparkan ke dalam kode Anda, itu menunjukkan strategi coba lagi aktif telah kehabisan batas yang dikonfigurasi seperti upaya maksimum atau coba lagi ember token. Lihat [the section called “Percobaan ulang”](#) bagian panduan ini untuk informasi lebih lanjut tentang strategi coba lagi. Lihat juga [the section called “Mengapa pengecualian dilemparkan sebelum mencapai upaya maksimal?”](#) topik?.

## Mengapa pengecualian dilemparkan sebelum mencapai upaya maksimal?

Terkadang Anda mungkin melihat pengecualian yang Anda harapkan akan dicoba lagi tetapi malah dilemparkan. Dalam situasi ini, langkah-langkah berikut dapat membantu menyelesaikan masalah.

- Verifikasi bahwa pengecualian dapat dicoba kembali. Beberapa pengecualian tidak dapat dicoba ulang, seperti yang menunjukkan permintaan layanan yang salah, kurangnya izin, dan sumber daya yang tidak ada, sebagai contoh. The SDK tidak secara otomatis mencoba kembali jenis pengecualian ini. Jika Anda menangkap pengecualian yang mewarisi dari `SdkBaseException`, Anda dapat memeriksa properti boolean `SdkBaseException.sdkErrorMetadata.isRetryable` untuk memverifikasi apakah SDK telah menentukan bahwa pengecualian dapat dicoba ulang.
- Verifikasi bahwa pengecualian dilemparkan ke kode Anda. Beberapa pengecualian muncul dalam pesan log sebagai informasi tetapi tidak benar-benar dilemparkan ke dalam kode Anda. Misalnya, pengecualian yang dapat dicoba ulang seperti kesalahan pelambatan mungkin dicatat karena SDK secara otomatis bekerja melalui beberapa siklus. `backoff-and-retry` Pemanggilan SDK operasi melempar pengecualian hanya jika tidak ditangani oleh pengaturan coba lagi yang dikonfigurasi.
- Verifikasi pengaturan coba ulang yang telah dikonfigurasi. Lihat [the section called “Percobaan ulang”](#) bagian panduan ini untuk informasi selengkapnya tentang strategi coba lagi dan kebijakan coba lagi. Pastikan kode Anda menggunakan pengaturan yang Anda harapkan atau default otomatis.
- Pertimbangkan untuk menyesuaikan pengaturan coba lagi Anda. Setelah Anda memverifikasi item sebelumnya, tetapi masalah tidak teratasi, Anda dapat mempertimbangkan untuk menyesuaikan pengaturan coba lagi.
  - Tingkatkan jumlah upaya maksimum. Secara default jumlah maksimum upaya untuk operasi adalah 3. Jika Anda menemukan bahwa ini tidak cukup dan pengecualian masih terjadi pada pengaturan default, pertimbangkan untuk meningkatkan `retryStrategy.maxAttempts` properti dalam konfigurasi klien Anda. Untuk informasi selengkapnya, lihat [the section called “Upaya maksimal”](#).
  - Tingkatkan pengaturan penundaan. Beberapa pengecualian mungkin dicoba ulang terlalu cepat sebelum kondisi yang mendasarinya memiliki kesempatan untuk diselesaikan. Jika Anda menduga hal itu terjadi, pertimbangkan untuk meningkatkan `retryStrategy.delayProvider.maxBackoff` properti `retryStrategy.delayProvider.initialDelay` atau dalam konfigurasi klien Anda. Untuk informasi selengkapnya, lihat [the section called “Penundaan dan backoff”](#).

- Nonaktifkan mode pemutus sirkuit. The SDK memelihara ember token untuk setiap klien layanan secara default. Ketika SDK mencoba permintaan dan gagal dengan pengecualian yang dapat dicoba ulang, jumlah token dikurangi; ketika permintaan berhasil, jumlah token bertambah.

Secara default, jika bucket token ini mencapai 0 token yang tersisa, sirkuit rusak. Setelah sirkuit rusak, percobaan ulang yang SDK dinonaktifkan dan permintaan saat ini dan selanjutnya yang gagal pada upaya pertama segera memberikan pengecualian. Percobaan SDK ulang yang diaktifkan kembali setelah upaya awal yang berhasil mengembalikan kapasitas yang cukup ke ember token. Perilaku ini disengaja dan dirancang untuk mencegah badai coba lagi selama pemadaman layanan dan pemulihan layanan.

Jika Anda lebih suka SDK terus mencoba ulang hingga upaya maksimum yang dikonfigurasi, pertimbangkan untuk menonaktifkan mode pemutus sirkuit dengan menyetel `retryStrategy.tokenBucket.useCircuitBreakerMode` properti ke `false` dalam konfigurasi klien Anda. Dengan properti ini disetel ke `false`, SDK klien menunggu hingga bucket token mencapai kapasitas yang cukup daripada meninggalkan percobaan ulang lebih lanjut yang mungkin menyebabkan pengecualian ketika ada 0 token yang tersisa.

## Bagaimana cara memperbaiki `NoSuchMethodError` atau `NoClassDefFoundError`?

SDK Bergantung pada berbagai AWS dependensi pihak ketiga untuk berfungsi dengan benar. Jika dependensi yang diharapkan tidak ada saat runtime atau merupakan versi yang tidak terduga, Anda mungkin melihat pengecualian runtime. `NoSuchMethodError`

Konflik ketergantungan biasanya terbagi dalam dua kategori: SDK /Konflik ketergantungan Smithy dan konflik ketergantungan pihak ketiga.

Saat membuat aplikasi Kotlin, biasanya Anda mengelola dependensi dengan Gradle. Menambahkan dependensi pada klien SDK layanan ke aplikasi Anda akan secara otomatis menyelesaikan dan menyertakan semua dependensi transitif. Jika aplikasi Anda memiliki dependensi lain, mereka mungkin bertentangan dengan dependensi yang dibutuhkan oleh SDK (misalnya, `OkHttp` adalah HTTP klien yang umum digunakan yang bergantung). SDK

Untuk mengatasi masalah tersebut, Anda mungkin perlu secara eksplisit menyelesaikan versi dependensi tertentu atau dependensi bayangan ke dalam namespace lokal untuk menghindari konflik. Resolusi ketergantungan Gradle adalah topik kompleks yang dibahas di bagian berikut dari Panduan Pengguna Kelas.



- [Memahami resolusi ketergantungan](#)
- [Kendala ketergantungan dan resolusi konflik](#)
- [Menyelaraskan versi ketergantungan](#)

## SDK/Konflik ketergantungan pandai besi

Secara umum, modul bergantung pada SDK modul lain dengan nomor versi yang sama. SDK Misalnya, `aws.sdk.kotlin:s3:1.2.3` tergantung pada `aws.sdk.kotlin:aws-http:1.2.3`, yang tergantung pada `aws.sdk.kotlin:aws-core:1.2.3`, dan sebagainya.

Selain itu, SDK modul bergantung pada versi modul Smithy yang spesifik dan terpadu juga. Nomor versi Smithy ini tidak sinkron dengan nomor SDK versi tetapi masih harus sesuai dengan versi yang diharapkan oleh SDK. Misalnya, `aws.sdk.kotlin:s3:1.2.3` mungkin tergantung pada `aws.smithy.kotlin:serde:1.1.1`, yang tergantung pada `aws.smithy.kotlin:runtime-core:1.1.1`, dan sebagainya.

Jika salah satu nomor versi ini tidak cocok, Anda mungkin mengalami konflik ketergantungan. Pastikan Anda memutakhirkan semua SDK dependensi Anda secara serempak dan tingkatkan dependensi Smithy eksplisit apa pun secara serempak juga. Pertimbangkan untuk menggunakan [katalog versi Gradle](#) kami untuk menjaga agar versi tetap sinkron dan menghilangkan pemetaan dugaan antara SDK dan versi Smithy. Lihat [the section called “Buat file build proyek”](#) topik untuk informasi dan contoh lebih lanjut.

[Selain itu, ketahuilah bahwa benjolan versi minor dalam modul SDK /Smithy mungkin berisi perubahan yang melanggar seperti yang dijelaskan dalam kebijakan pembuatan versi. SDK](#) Saat memutakhirkan di antara versi minor, berhati-hatilah untuk memeriksa changelog dan memvalidasi perilaku runtime secara menyeluruh.

## Saya melihat **NoClassDefFoundError** untuk **okhttp3/coroutines/ExecuteAsyncKt**

Jika Anda melihat kesalahan ini, kemungkinan besar berarti Anda belum mengonfigurasi klien layanan Anda untuk menggunakan `OkHttp4Engine`. [Tinjau dokumentasi](#) tentang cara mengonfigurasi Gradle dan gunakan `OkHttp4Engine` dalam kode Anda.

# Bekerja dengan Layanan AWS menggunakan AWS SDK for Kotlin

Bab ini berisi informasi tentang cara bekerja Layanan AWS dengan menggunakan SDK for Kotlin.

## Daftar Isi

- [Bekerja dengan Amazon S3 menggunakan AWS SDK for Kotlin](#)
  - [Perlindungan integritas data dengan checksum](#)
    - [Mengunggah objek](#)
      - [Gunakan nilai checksum yang telah dihitung sebelumnya](#)
      - [Unggahan multipart](#)
    - [Unduh objek](#)
      - [Validasi asinkron](#)
  - [Bekerja dengan Titik Akses Multi-Wilayah Amazon S3 dengan menggunakan for Kotlin SDK](#)
    - [Bekerja dengan Titik Akses Multi-Wilayah](#)
    - [Bekerja dengan objek dan Titik Akses Multi-Wilayah](#)
- [Bekerja dengan DynamoDB menggunakan AWS SDK for Kotlin](#)
  - [Gunakan AWS titik akhir berbasis akun](#)
  - [Memetakan kelas ke item DynamoDB dengan menggunakan DynamoDB Mapper \(Pratinjau Pengembang\)](#)
    - [Memulai dengan DynamoDB Mapper](#)
      - [Tambahkan dependensi](#)
      - [Membuat dan menggunakan mapper](#)
      - [Tentukan skema dengan anotasi kelas](#)
      - [Memanggil operasi](#)
        - [Bekerja dengan tanggapan berhalaman](#)
    - [Konfigurasi DynamoDB Mapper](#)
      - [Gunakan pencegat](#)
        - [Memahami pipa permintaan](#)
        - [Kait](#)
          - [Kait hanya-baca](#)

- [Memodifikasi kait](#)
- [Perintah eksekusi](#)
- [Contoh konfigurasi](#)
- [Hasilkan skema dari anotasi](#)
  - [Terapkan plugin](#)
  - [Konfigurasikan plugin](#)
  - [Menganotasi kelas](#)
    - [Anotasi kelas](#)
    - [Anotasi properti](#)
  - [Tentukan konverter item khusus](#)
- [Tentukan skema secara manual](#)
  - [Tentukan skema dalam kode](#)
- [Gunakan indeks sekunder dengan DynamoDB Mapper](#)
  - [Tentukan skema untuk indeks sekunder](#)
  - [Gunakan indeks sekunder dalam operasi](#)
- [Gunakan ekspresi](#)
  - [Gunakan ekspresi dalam operasi](#)
  - [Komponen-komponen DSL](#)
    - [Atribut](#)
    - [Kesetaraan dan ketidaksetaraan](#)
    - [Rentang dan set](#)
    - [Logika Boolean](#)
    - [Fungsi dan properti](#)
    - [Urutkan filter kunci](#)

## Bekerja dengan Amazon S3 menggunakan AWS SDK for Kotlin

Antarmuka utama Anda ke Amazon Simple Storage Service untuk Kotlin SDK adalah [S3Client](#). Gunakan `S3Client` seperti klien layanan lainnya di SDK untuk membuat [permintaan](#) ke Amazon S3.

Sumber daya untuk membantu Anda menggunakan Kotlin SDK dengan S3 adalah:

- [SDK API referensi Kotlin untuk S3](#).
- [Panduan Pengguna layanan S3](#) dan [API referensi layanan](#).

Topik berikut menyajikan contoh kode terpandu untuk Kotlin tertentu SDK APIs yang bekerja dengan S3.

Topik

- [Perlindungan integritas data dengan checksum](#)
- [Bekerja dengan Titik Akses Multi-Wilayah Amazon S3 dengan menggunakan for Kotlin SDK](#)

## Perlindungan integritas data dengan checksum

Amazon Simple Storage Service (Amazon S3) menyediakan kemampuan untuk menentukan checksum saat Anda mengunggah objek. Ketika Anda menentukan checksum, itu disimpan dengan objek dan dapat divalidasi ketika objek diunduh.

Checksum menyediakan lapisan integritas data tambahan saat Anda mentransfer file. Dengan checksum, Anda dapat memverifikasi konsistensi data dengan mengonfirmasi bahwa file yang diterima cocok dengan file asli. [Untuk informasi selengkapnya tentang checksum dengan Amazon S3, lihat Panduan Pengguna Layanan Penyimpanan Sederhana Amazon termasuk algoritme yang didukung.](#)

Anda memiliki fleksibilitas untuk memilih algoritma yang paling sesuai dengan kebutuhan Anda dan membiarkan SDK menghitung checksum. Atau, Anda dapat memberikan nilai checksum yang telah dihitung sebelumnya dengan menggunakan salah satu algoritme yang didukung.

### Note

Dimulai dengan versi 1.4.0 AWS SDK for Kotlin, SDK menyediakan perlindungan integritas default dengan menghitung CRC32 checksum secara otomatis untuk unggahan. SDK menghitung checksum ini jika Anda tidak memberikan nilai checksum yang telah dihitung sebelumnya atau jika Anda tidak menentukan algoritme yang harus digunakan SDK untuk menghitung checksum.

SDK juga menyediakan pengaturan global untuk perlindungan integritas data yang dapat Anda atur secara eksternal, yang dapat Anda baca di Panduan Referensi Alat [AWS SDKs dan Alat](#).

Kami membahas checksum dalam dua fase permintaan: mengunggah objek dan mengunduh objek.

## Mengunggah objek

Anda mengunggah objek ke Amazon S3 dengan SDK untuk Kotlin dengan menggunakan `putObject` fungsi dengan parameter permintaan. Jenis data permintaan menyediakan `checksumAlgorithm` properti untuk mengaktifkan perhitungan checksum.

Cuplikan kode berikut menunjukkan permintaan untuk mengunggah objek dengan checksum. CRC32 Ketika SDK mengirim permintaan, ia menghitung CRC32 checksum dan mengunggah objek. Amazon S3 menyimpan checksum dengan objek.

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

Jika Anda tidak menyediakan algoritma checksum dengan permintaan, perilaku checksum bervariasi tergantung pada versi SDK yang Anda gunakan seperti yang ditunjukkan pada tabel berikut.

Perilaku checksum ketika tidak ada algoritma checksum yang disediakan

Versi SDK Kotlin	Perilaku checksum
lebih awal dari 1.4.0	SDK tidak secara otomatis menghitung checksum berbasis CRC dan menyediakannya dalam permintaan.
1.4.0 atau yang lebih baru	SDK menggunakan CRC32 algoritma untuk menghitung checksum dan menyediakannya dalam permintaan. Amazon S3 memvalidasi integritas transfer dengan menghitung checksumnya sendiri dan membandingkannya dengan CRC32 checksum yang disediakan oleh SDK. Jika checksum cocok, checksum disimpan dengan objek.

## Gunakan nilai checksum yang telah dihitung sebelumnya

Nilai checksum yang telah dihitung sebelumnya yang disertakan dengan permintaan menonaktifkan komputasi otomatis oleh SDK dan menggunakan nilai yang disediakan sebagai gantinya.

Contoh berikut menunjukkan permintaan dengan SHA256 checksum yang telah dihitung sebelumnya.

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

Jika Amazon S3 menentukan nilai checksum salah untuk algoritme yang ditentukan, layanan akan mengembalikan respons kesalahan.

## Unggahan multipart

Anda juga dapat menggunakan checksum dengan unggahan multipart.

Anda harus menentukan algoritma checksum dalam `CreateMultipartUpload` permintaan dan di setiap `UploadPart` permintaan. Sebagai langkah terakhir, Anda harus menentukan checksum dari setiap bagian di `CompleteMultipartUpload`. Contoh berikut menunjukkan cara membuat upload multipart dengan algoritma checksum yang ditentukan.

```
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 = ByteString.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
    }
}
}

```

## Unduh objek

Saat Anda menggunakan metode [getObject](#) untuk mengunduh objek, SDK secara otomatis memvalidasi checksum nilai kuncinya. `ChecksumMode.enabled` ketika `checksumMode` properti pembangun untuk `GetObjectRequest` diatur ke `ChecksumMode.Enabled`.

Permintaan dalam cuplikan berikut mengarahkan SDK untuk memvalidasi checksum dalam respons dengan menghitung checksum dan membandingkan nilainya.

```

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

```

Jika objek tidak diunggah dengan checksum, tidak ada validasi yang terjadi.

Jika Anda menggunakan versi SDK 1.4.0 atau yang lebih baru, SDK secara otomatis memeriksa integritas `getObject` permintaan tanpa `checksumMode = ChecksumMode.Enabled` menambahkan permintaan.

## Validasi asinkron

Karena SDK untuk Kotlin menggunakan respons streaming saat mengunduh objek dari Amazon S3, checksum akan dihitung saat Anda menggunakan objek. Oleh karena itu, Anda harus mengonsumsi objek sehingga checksum divalidasi.

Contoh berikut menunjukkan cara memvalidasi checksum dengan sepenuhnya mengonsumsi respon.

```
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.
}
```

Sebaliknya, kode dalam contoh berikut tidak menggunakan objek dengan cara apa pun, sehingga checksum tidak divalidasi.

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

Jika checksum yang dihitung oleh SDK tidak cocok dengan checksum yang diharapkan yang dikirim dengan respons, SDK akan melempar `ChecksumMismatchException`

## Bekerja dengan Titik Akses Multi-Wilayah Amazon S3 dengan menggunakan for Kotlin SDK

Titik Akses Multi-Wilayah Amazon S3 menyediakan titik akhir global yang dapat digunakan aplikasi untuk memenuhi permintaan dari bucket Amazon S3 yang terletak di beberapa tempat. Wilayah AWS Anda dapat menggunakan Titik Akses Multi-Wilayah untuk membangun aplikasi Multi-wilayah dengan arsitektur yang sama yang digunakan di satu Wilayah, dan kemudian menjalankan aplikasi tersebut di mana saja di dunia.

Panduan Pengguna Amazon S3 berisi informasi latar belakang selengkapnya tentang [Titik Akses Multi-Wilayah](#).



## Bekerja dengan Titik Akses Multi-Wilayah

Untuk membuat Titik Akses Multi-Wilayah, mulailah dengan menentukan satu bucket di setiap AWS Wilayah yang ingin Anda layani permintaan. Cuplikan berikut membuat dua ember.

Buat ember

Fungsi berikut membuat dua bucket untuk bekerja dengan Multi-Region Access Point. Satu ember ada di Wilayah us-east-1 dan yang lainnya di Wilayah us-west-1.

Pembuatan klien S3 yang diteruskan sebagai argumen pertama ditampilkan pada contoh pertama di bawah [the section called "Bekerja dengan benda"](#).

```
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.")
    }
}

```

Anda menggunakan [klien kontrol S3](#) Kotlin SDK untuk membuat, menghapus, dan mendapatkan informasi tentang Titik Akses Multi-Wilayah.

Tambahkan ketergantungan pada artefak kontrol S3 seperti yang ditunjukkan pada cuplikan berikut. (Anda dapat menavigasi ke [X.Y.Z](#) tautan untuk melihat versi terbaru yang tersedia.)

```

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

```

Konfigurasi klien kontrol S3 untuk bekerja dengan Wilayah AWS us-west-2 seperti yang ditunjukkan pada kode berikut. Semua operasi klien kontrol S3 harus menargetkan us-west-2 wilayah tersebut.

```

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

```

Gunakan klien kontrol S3 untuk membuat Titik Akses Multi-Wilayah dengan menentukan nama bucket (dibuat sebelumnya) seperti yang ditunjukkan pada kode berikut.

```

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"
}

```

Karena pembuatan Titik Akses Multi-Wilayah adalah operasi asinkron, Anda menggunakan token yang Anda terima dari respons langsung untuk memeriksa status proses pembuatan. Setelah pemeriksaan status mengembalikan pesan sukses, Anda dapat menggunakan `GetMultiRegionAccessPoint` operasi untuk mendapatkan alias Titik Akses Multi-Wilayah. Alias adalah komponen terakhir dari ARN, yang Anda butuhkan untuk operasi tingkat objek.

## Gunakan token untuk memeriksa status

Gunakan `DescribeMultiRegionAccessPointOperation` untuk memeriksa status operasi terakhir. Setelah `requestStatus` nilainya menjadi "SUCCEEDED", Anda dapat bekerja dengan Titik Akses Multi-Wilayah.

```
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)
    }
}
```

## Bekerja dengan objek dan Titik Akses Multi-Wilayah

Anda menggunakan [klien S3](#) untuk bekerja dengan objek di Multi-Region Access Points. Banyak operasi yang Anda gunakan pada objek dalam ember yang dapat Anda gunakan pada Titik Akses Multi-Wilayah. Untuk informasi selengkapnya dan daftar lengkap operasi, lihat [Kompatibilitas Titik Akses Multi-Wilayah dengan operasi S3](#).

Operasi dengan Titik Akses Multi-Wilayah ditandatangani dengan algoritma penandatanganan Asymmetric Sigv4 (Sigv4a). Support untuk Sigv4a AWS SDK for Kotlin saat ini memerlukan penandatanganan dengan CRT penandatanganan yang merupakan dependensi terpisah. Untuk mengonfigurasi dukungan Sigv4a, tambahkan dependensi berikut ke proyek Anda. (Anda dapat menavigasi ke [X.Y.Z](#) tautan untuk melihat versi terbaru yang tersedia.)

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

Setelah Anda menambahkan dependensi, konfigurasi klien S3 untuk menggunakan algoritma penandatanganan Sigv4a seperti yang ditunjukkan pada kode berikut.

```
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
}
```

Setelah Anda mengonfigurasi klien S3, operasi yang didukung S3 untuk Titik Akses Multi-Wilayah bekerja sama. Satu-satunya perbedaan adalah bahwa parameter bucket harus ARN dari Multi-Region Access Point. Anda bisa mendapatkan ARN dari konsol Amazon S3 atau secara terprogram seperti yang ditunjukkan sebelumnya dalam `createMrap` fungsi yang mengembalikan file. ARN

Contoh kode berikut menunjukkan yang ARN digunakan dalam `GetObject` operasi.

```
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
}
```

## Bekerja dengan DynamoDB menggunakan AWS SDK for Kotlin

### Gunakan AWS titik akhir berbasis akun

DynamoDB [AWS menawarkan endpoint berbasis akun](#) yang dapat meningkatkan kinerja dengan menggunakan ID akun AWS Anda untuk merampungkan perutean permintaan.

Untuk memanfaatkan fitur ini, Anda perlu menggunakan versi 1.3.37 atau lebih besar. AWS SDK for Kotlin Anda dapat menemukan versi terbaru dari yang SDK tercantum di repositori [pusat Maven](#). Setelah versi yang SDK didukung aktif, secara otomatis menggunakan titik akhir baru.

Jika Anda ingin keluar dari perutean berbasis akun, Anda memiliki empat opsi:

- Konfigurasi klien layanan DynamoDB dengan `AccountIdEndpointMode` set ke `DISABLED`
- Tetapkan variabel lingkungan.
- Tetapkan properti JVM sistem.
- Perbarui pengaturan file AWS konfigurasi bersama.


Cuplikan berikut adalah contoh cara menonaktifkan routing berbasis akun dengan mengonfigurasi klien layanan DynamoDB:

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

```
}
```

Panduan Referensi AWS SDKs and Tools memberikan informasi lebih lanjut tentang [tiga opsi konfigurasi](#) terakhir.

## Memetakan kelas ke item DynamoDB dengan menggunakan DynamoDB Mapper (Pratinjau Pengembang)

 DynamoDB Mapper adalah rilis Pratinjau Pengembang. Ini tidak lengkap fitur dan dapat berubah.


[DynamoDB Mapper](#) adalah [library tingkat tinggi yang menawarkan mekanisme untuk memetakan class Kotlin ke tabel dan indeks DynamoDB, mirip dengan DynamoDB Enhanced Client atau Object AWS SDK for Java Persistence Model. AWS SDK for .NET](#)

Anda menentukan skema yang menggambarkan objek data Anda dan cara mengonversinya menjadi item DynamoDB. Setelah Anda menentukan skema, DynamoDB Mapper menyediakan antarmuka intuitif untuk menggunakan objek Anda dalam membuat, membaca, memperbarui, atau menghapus CRUD () operasi pada tabel dan indeks Anda.

### Topik

- [Memulai dengan DynamoDB Mapper](#)
- [Konfigurasi DynamoDB Mapper](#)
- [Hasilkan skema dari anotasi](#)
- [Tentukan skema secara manual](#)
- [Gunakan indeks sekunder dengan DynamoDB Mapper](#)
- [Gunakan ekspresi](#)

## Memulai dengan DynamoDB Mapper

 DynamoDB Mapper adalah rilis Pratinjau Pengembang. Ini tidak lengkap fitur dan dapat berubah.

Tutorial berikut memperkenalkan komponen dasar DynamoDB Mapper dan menunjukkan bagaimana menggunakannya dalam kode Anda.

Tambahkan dependensi

Untuk mulai bekerja dengan DynamoDB Mapper dalam proyek Gradle Anda, tambahkan plugin dan dua dependensi ke file `Anda.build.gradle.kts`

(Anda dapat menavigasi ke [X.Y.Z](#) tautan untuk melihat versi terbaru yang tersedia.)

```
// 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")
}
```

\*Ganti *<Version>* dengan rilis SDK terbaru. Untuk menemukan versi terbaru SDK, periksa [rilis terbaru di GitHub](#).

#### Note

Beberapa dependensi ini bersifat opsional jika Anda berencana untuk mendefinisikan skema secara manual. Lihat [the section called "Tentukan skema secara manual"](#) untuk informasi lebih lanjut dan kumpulan dependensi yang dikurangi.

Membuat dan menggunakan mapper

DynamoDB Mapper menggunakan klien DynamoDB untuk AWS SDK for Kotlin berinteraksi dengan DynamoDB. Anda perlu menyediakan instance yang sepenuhnya dikonfigurasi saat membuat [DynamoDbClient](#) instance mapper seperti yang ditunjukkan pada cuplikan kode berikut:

```
import aws.sdk.kotlin.hll.dynamodbmapper.DynamoDbMapper
```



```
import aws.sdk.kotlin.services.dynamodb.DynamoDbClient

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

Setelah Anda membuat instance mapper, Anda dapat menggunakannya untuk mendapatkan contoh tabel seperti yang ditunjukkan berikut:

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

Kode sebelumnya mendapat referensi ke tabel DynamoDB bernama `cars` dengan skema yang ditentukan oleh `CarSchema` (kita membahas skema di bawah). Setelah Anda membuat instance tabel, Anda dapat melakukan operasi terhadapnya. Cuplikan kode berikut menunjukkan dua contoh operasi terhadap tabel `cars`:

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

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

Kode sebelumnya membuat item baru dalam `cars` tabel. Kode membuat `Car` instance inline menggunakan `Car` kelas, yang definisinya ditunjukkan di bawah ini. Selanjutnya, kode menanyakan `cars` tabel untuk item yang kunci partisi Peugeot dan mencetaknya. Operasi [dijelaskan secara lebih rinci di bawah ini](#).

### Tentukan skema dengan anotasi kelas

Untuk berbagai class Kotlin, SDK dapat secara otomatis menghasilkan skema pada waktu pembuatan dengan menggunakan plugin DynamoDB Mapper Schema Generator untuk Gradle. Saat Anda menggunakan generator skema, SDK akan memeriksa kelas Anda untuk menyimpulkan skema, yang mengurangi beberapa boilerplate yang terlibat dalam mendefinisikan skema secara manual. [Anda dapat menyesuaikan skema yang dihasilkan dengan menggunakan anotasi dan konfigurasi tambahan](#).

Untuk menghasilkan skema dari anotasi, pertama-tama beri anotasi pada kelas Anda dengan dan kunci apa pun dengan [@DynamoDbItem](#) dan [@DynamoDbPartitionKey](#) [@DynamoDbSortKey](#). Kode berikut menunjukkan kelas beranotasiCar:

```
// 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
)
```

Setelah membangun, Anda dapat merujuk ke yang dihasilkan secara otomatisCarSchema. Anda dapat menggunakan referensi dalam getTable metode mapper untuk mendapatkan contoh tabel seperti yang ditunjukkan dalam berikut ini:

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

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

Atau, Anda bisa mendapatkan instance tabel dengan memanfaatkan metode ekstensi [DynamoDbMapper](#) yang dibuat secara otomatis pada waktu pembuatan. Dengan menggunakan pendekatan ini, Anda tidak perlu merujuk ke skema dengan nama. Seperti yang ditunjukkan dalam berikut ini, metode getCarsTable ekstensi yang dihasilkan secara otomatis mengembalikan referensi ke contoh tabel:

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

Lihat [the section called “Menghasilkan skema”](#) untuk detail dan contoh selengkapnya.

## Memanggil operasi

DynamoDB Mapper mendukung subset operasi yang tersedia di SDK. DynamoDbClient Operasi mapper diberi nama sama dengan rekan-rekan mereka pada klien SDK. Banyak anggota permintaan/

respons mapper sama dengan rekan klien SDK mereka, meskipun beberapa telah diganti namanya, diketik ulang, atau dihapus sama sekali.

Anda memanggil operasi pada instance tabel menggunakan sintaks DSL seperti yang ditunjukkan dalam berikut ini:

```
import aws.sdk.kotlin.h11.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)
```

Anda juga dapat memanggil operasi dengan menggunakan objek permintaan eksplisit:

```
import aws.sdk.kotlin.h11.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)
```

Dua contoh kode sebelumnya setara.

### Bekerja dengan tanggapan berhalaman

Beberapa operasi seperti query dan scan dapat mengembalikan koleksi data yang mungkin terlalu besar untuk dikembalikan dalam satu respons. Untuk memastikan bahwa semua objek diproses, DynamoDB Mapper menyediakan metode paginating, yang tidak segera memanggil DynamoDB, tetapi mengembalikan tipe respons operasi, seperti [Flow](#) yang ditunjukkan dalam berikut ini:

```
Flow<ScanResponse<Car>>
```

```
import aws.sdk.kotlin.h11.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) }
}
```


Seringkali, aliran objek lebih berguna untuk logika bisnis daripada aliran respons yang mengandung objek. Mapper menyediakan metode ekstensi pada respons paginasi untuk mengakses aliran objek. Misalnya, kode berikut mengembalikan `Flow<Car>` bukan `Flow<ScanResponse<Car>>` seperti yang ditunjukkan sebelumnya:

```
import aws.sdk.kotlin.h11.dynamodbmapper.operations.items
import aws.sdk.kotlin.h11.dynamodbmapper.operations.scanPaginated

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

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

## Konfigurasi DynamoDB Mapper

 DynamoDB Mapper adalah rilis Pratinjau Pengembang. Ini tidak lengkap fitur dan dapat berubah.

DynamoDB Mapper menawarkan opsi konfigurasi yang dapat Anda gunakan menyesuaikan perilaku perpustakaan agar sesuai dengan aplikasi Anda.

### Gunakan pencegat

Pustaka DynamoDB Mapper mendefinisikan kait yang dapat Anda manfaatkan pada tahap kritis dari pipeline permintaan mapper. Anda dapat mengimplementasikan [Interceptor](#) antarmuka untuk mengimplementasikan kait untuk mengamati atau memodifikasi proses mapper.

Anda dapat mendaftarkan satu atau lebih pencegat pada satu DynamoDB Mapper sebagai opsi konfigurasi. Lihat [contoh](#) di akhir bagian ini untuk bagaimana Anda mendaftarkan pencegat.

## Memahami pipa permintaan

Pipeline permintaan mapper terdiri dari 5 langkah berikut:

1. Inisialisasi: Mengatur operasi dan mengumpulkan konteks awal.
2. Serialisasi: Ubah objek permintaan tingkat tinggi menjadi objek permintaan tingkat rendah. Langkah ini mengubah objek Kotlin tingkat tinggi menjadi item DynamoDB yang terdiri dari nama dan nilai atribut.
3. Pemanggilan tingkat rendah: Jalankan permintaan pada klien DynamoDB yang mendasarinya.
4. Deserialisasi: Ubah objek respons tingkat rendah menjadi objek respons tingkat tinggi. Langkah ini mencakup mengonversi item DynamoDB yang terdiri dari nama dan nilai atribut menjadi objek Kotlin tingkat tinggi.
5. Penyelesaian: Selesaikan respons tingkat tinggi untuk kembali ke penelepon. Jika pengecualian dilemparkan selama eksekusi pipeline, langkah ini menyelesaikan pengecualian yang dilemparkan ke pemanggil.

## Kait

Hooks adalah metode pencegat yang dipanggil mapper sebelum atau sesudah langkah-langkah tertentu dalam pipeline. Ada dua varian kait: `read-only` dan `modify` (atau `read-write`). Misalnya, `readBeforeInvocation` adalah hook hanya-baca yang dieksekusi oleh mapper dalam fase sebelum langkah pemanggilan tingkat rendah.

### Kait hanya-baca

Pemeta memanggil kait hanya-baca sebelum dan sesudah setiap langkah dalam pipeline (kecuali sebelum langkah Inisialisasi dan setelah langkah Penyelesaian). Tudung hanya-baca menawarkan tampilan hanya-baca dari operasi tingkat tinggi yang sedang berlangsung. Mereka menyediakan mekanisme untuk memeriksa keadaan operasi untuk logging, debugging, mengumpulkan metrik, misalnya. Setiap hook `read-only` menerima argumen konteks dan kembali. [Unit](#)

Mapper menangkap pengecualian apa pun yang dilemparkan selama hook hanya-baca dan menambahkannya ke konteks. Kemudian melewati konteks dengan pengecualian ke kait pencegat berikutnya dalam fase yang sama. Pemeta melempar pengecualian apa pun ke pemanggil hanya setelah memanggil hook hanya-baca pencegat terakhir untuk fase yang sama. Misalnya, jika mapper dikonfigurasi dengan dua A pencegat dan B, dan A `readAfterSerialization` hook melempar pengecualian, mapper menambahkan pengecualian ke konteks yang diteruskan ke hook.

`readAfterSerialization` Setelah `readAfterSerialization` hook selesai, mapper melempar pengecualian kembali ke pemanggil.

## Memodifikasi kait

Pemeta memanggil kait modifikasi sebelum setiap langkah dalam pipeline (kecuali sebelum Inisialisasi). `Modify` hook menawarkan kemampuan untuk melihat dan memodifikasi operasi tingkat tinggi yang sedang berlangsung. Mereka dapat digunakan untuk menyesuaikan perilaku dan data dengan cara yang konfigurasi mapper dan skema item tidak. Setiap hook modifikasi menerima argumen konteks dan mengembalikan beberapa subset dari konteks itu sebagai hasil—baik dimodifikasi oleh hook atau dilewati dari konteks input.

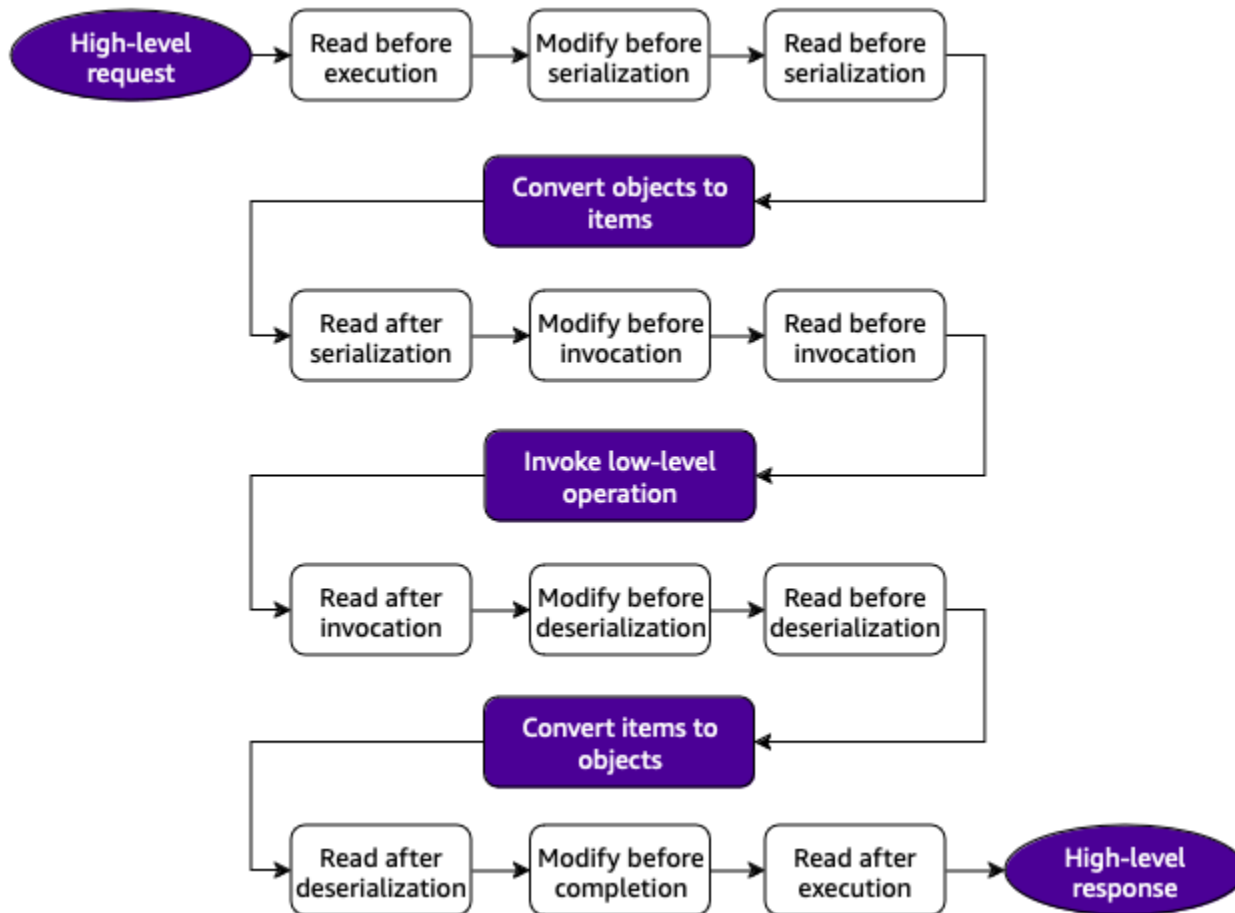
Jika mapper menangkap pengecualian saat mengeksekusi hook modifikasi, itu tidak mengeksekusi kait modifikasi pencegat lain dalam fase yang sama. Pemeta menambahkan pengecualian ke konteks dan meneruskannya ke hook hanya-baca berikutnya. Pemeta melempar pengecualian apa pun ke pemanggil hanya setelah memanggil hook hanya-baca pencegat terakhir untuk fase yang sama. Misalnya, jika mapper dikonfigurasi dengan dua A pencegat dan B, dan A `modifyBeforeSerialization` hook melempar pengecualian, B `modifyBeforeSerialization` hook tidak akan dipanggil. `Interceptors` dan A B' s `readAfterSerialization` hook akan mengeksekusi, setelah itu pengecualian akan dilemparkan kembali ke penelepon.

## Perintah eksekusi

Urutan pencegat didefinisikan dalam konfigurasi mapper menentukan urutan pemetaan memanggil kait:

- Untuk fase sebelum langkah pemanggilan tingkat rendah, ia mengeksekusi kait dalam urutan yang sama dengan yang ditambahkan dalam konfigurasi.
- Untuk fase setelah langkah pemanggilan tingkat rendah, ia mengeksekusi kait dalam urutan terbalik dari urutan yang ditambahkan dalam konfigurasi.

Diagram berikut menunjukkan urutan eksekusi metode hook:



Deskripsi teks dari urutan eksekusi metode hook

Seorang mapper mengeksekusi kait pencegat dalam urutan sebagai berikut:

1. DynamoDB Mapper memanggil permintaan tingkat tinggi
2. Baca sebelum eksekusi
3. Ubah sebelum serialisasi
4. Baca sebelum serialisasi
5. DynamoDB Mapper mengkonversi objek ke item
6. Baca setelah serialisasi
7. Ubah sebelum pemanggilan
8. Baca sebelum pemanggilan
9. DynamoDB Mapper memanggil operasi tingkat rendah
10. Baca setelah doa
11. Ubah sebelum deserialisasi

- 12Baca sebelum deserialisasi
- 13DynamoDB Mapper mengkonversi item ke objek
- 14Baca setelah deserialisasi
- 15Ubah sebelum selesai
- 16Baca setelah eksekusi
- 17DynamoDB Mapper mengembalikan respons tingkat tinggi

## Contoh konfigurasi

Contoh berikut menunjukkan cara mengkonfigurasi pencegat pada sebuah DynamoDbMapper instance:


```
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
}
```

## Hasilkan skema dari anotasi

 DynamoDB Mapper adalah rilis Pratinjau Pengembang. Ini tidak lengkap fitur dan dapat berubah.



DynamoDB Mapper bergantung pada skema yang menentukan pemetaan antara class Kotlin dan item DynamoDB. Class Kotlin Anda dapat mendorong pembuatan skema dengan menggunakan plugin Gradle generator skema.

### Terapkan plugin

Untuk memulai skema pembuatan kode untuk kelas Anda, terapkan plugin dalam skrip build aplikasi Anda dan tambahkan dependensi pada modul anotasi. Cuplikan skrip Gradle berikut menunjukkan penyiapan yang diperlukan untuk pembuatan kode.

(Anda dapat menavigasi ke [X.Y.Z](#) tautan untuk melihat versi terbaru yang tersedia.)

```
// 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")
}
```

### Konfigurasi plugin

Plugin ini menawarkan sejumlah opsi konfigurasi yang dapat Anda terapkan dengan menggunakan ekstensi `dynamoDbMapper { ... }` plugin di skrip build Anda:

Opsi	Deskripsi opsi	Nilai
<code>generateBuilderClasses</code>	Mengontrol apakah kelas pembangun DSL -style akan dibuat untuk kelas yang dianotasi <code>@DynamoDbItem</code>	<p>WHEN_REQUIRED (default): Kelas pembangun tidak akan dibuat untuk kelas yang hanya terdiri dari anggota publik yang dapat berubah dan memiliki konstruktor zero-arg</p> <p>ALWAYS: Kelas pembangun akan selalu dihasilkan</p>

Opsi	Deskripsi opsi	Nilai
<code>visibility</code>	Mengontrol visibilitas kelas yang dihasilkan	PUBLIC (default) INTERNAL
<code>destinationPackage</code>	Menentukan nama paket untuk kelas yang dihasilkan	RELATIVE(default): Kelas skema akan dihasilkan dalam sub-paket relatif terhadap kelas beranotasi Anda. Secara default, sub-paket diberi namadynamodbmapper.generatedschemas, dan ini dapat dikonfigurasi dengan melewati parameter string  ABSOLUTE: Kelas skema akan dihasilkan dalam paket absolut relatif terhadap root aplikasi Anda. Secara default, paket diberi namaaws.sdk.kotlin.hll.dynamodbmapper.generatedschemas, dan ini dapat dikonfigurasi dengan melewati parameter string.
<code>generateGetTableExtension</code>	Mengontrol apakah metode <code>DynamoDbMapper.getTableExtension</code> akan dihasilkan	true (default) false

Example Contoh konfigurasi plugin pembuatan kode

Contoh berikut ini mengonfigurasi paket tujuan dan visibilitas skema yang dihasilkan:

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

```
import aws.sdk.kotlin.h11.dynamodbmapper.codegen.annotations.DestinationPackage
import aws.sdk.kotlin.h11.dynamodbmapper.codegen.annotations.Visibility
import aws.smithy.kotlin.runtime.ExperimentalApi

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

## Menganotasi kelas

Generator skema mencari anotasi kelas untuk menentukan kelas mana yang akan menghasilkan skema. Untuk ikut serta dalam membuat skema, beri anotasi pada kelas Anda. [@DynamoDbItem](#) Anda juga harus membuat anotasi properti kelas yang berfungsi sebagai kunci partisi item dengan anotasi [@DynamoDbPartitionKey](#).

Definisi kelas berikut menunjukkan anotasi minimal yang diperlukan untuk pembuatan skema:

### Example

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

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

## Anotasi kelas

Anotasi berikut diterapkan ke kelas untuk mengontrol pembuatan skema:

- [@DynamoDbItem](#): Menentukan bahwa kelas/antarmuka ini menjelaskan jenis item dalam tabel. Semua properti publik dari jenis ini akan dipetakan ke atribut kecuali mereka secara eksplisit diabaikan. Saat hadir, skema akan dihasilkan untuk kelas ini.
- `converterName`: Parameter opsional yang menunjukkan skema khusus harus digunakan daripada yang dibuat oleh plugin generator skema. Ini adalah nama yang sepenuhnya memenuhi syarat dari `ItemConverter` kelas kustom. [the section called “Tentukan konverter item khusus”](#) Bagian ini menunjukkan contoh membuat dan menggunakan skema kustom.

## Anotasi properti

Anda dapat menerapkan anotasi berikut ke properti kelas untuk mengontrol pembuatan skema:

- [@DynamoDbPartitionKey](#): Menentukan kunci partisi untuk item.
- [@DynamoDbSortKey](#): Menentukan kunci sortir opsional untuk item.
- [@DynamoDbIgnore](#): Menentukan bahwa properti kelas ini tidak harus dikonversi ke/dari atribut Item oleh DynamoDB Mapper.
- [@DynamoDbAttribute](#): Menentukan nama atribut kustom opsional untuk properti kelas ini.

## Tentukan konverter item khusus

Dalam beberapa kasus, Anda mungkin ingin menentukan konverter item khusus untuk kelas Anda. Salah satu alasannya adalah jika kelas Anda menggunakan tipe yang tidak didukung oleh plugin generator skema. Kami menggunakan versi Employee kelas berikut sebagai contoh:

```
import kotlin.uuid.Uuid

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

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

EmployeeKelas sekarang menggunakan `kotlin.uuid.Uuid` tipe, yang saat ini tidak didukung oleh generator skema. Pembuatan skema gagal dengan kesalahan: `Unsupported attribute type TypeRef(pkg=kotlin.uuid, shortName=Uuid, genericArgs=[], nullable=false)`. Kesalahan ini menunjukkan bahwa plugin tidak dapat menghasilkan konverter item untuk kelas ini. Karena itu, kita perlu menulis sendiri.

Untuk melakukan ini, kami menerapkan [ItemConverter](#) untuk kelas, kemudian memodifikasi anotasi `@DynamoDbItem` kelas dengan menentukan nama yang sepenuhnya memenuhi syarat dari konverter item baru.

Pertama, kami menerapkan a [ValueConverter](#) untuk `kotlin.uuid.Uuid` kelas:

```
import aws.sdk.kotlin.h11.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())
}
```

Kemudian, kami menerapkan `ItemConverter` untuk `Employee` kelas kami.

`ItemConverter` menggunakan konverter nilai baru ini dalam deskripsi atribut untuk "workstationId":

```
import aws.sdk.kotlin.h11.dynamodbmapper.items.AttributeDescriptor
import aws.sdk.kotlin.h11.dynamodbmapper.items.ItemConverter
import aws.sdk.kotlin.h11.dynamodbmapper.items.SimpleItemConverter
import aws.sdk.kotlin.h11.dynamodbmapper.values.scalars.IntConverter
import aws.sdk.kotlin.h11.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
        )
    ),
)

```

Sekarang kita telah mendefinisikan konverter item, kita dapat menerapkannya ke kelas kita. Kami memperbarui `@DynamoDbItem` anotasi untuk mereferensikan konverter item dengan memberikan nama kelas yang sepenuhnya memenuhi syarat seperti yang ditunjukkan pada berikut ini:

```

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
)

```

Akhirnya kita bisa mulai menggunakan kelas dengan DynamoDB Mapper.

## Tentukan skema secara manual

 DynamoDB Mapper adalah rilis Pratinjau Pengembang. Ini tidak lengkap fitur dan dapat berubah.

### Tentukan skema dalam kode

Untuk kontrol dan penyesuaian maksimum, Anda dapat secara manual menentukan dan menyesuaikan skema dalam kode.

Seperti yang ditunjukkan dalam cuplikan berikut, Anda perlu menyertakan lebih sedikit dependensi dalam `build.gradle.kts` file Anda dibandingkan dengan menggunakan pembuatan skema berbasis anotasi.

(Anda dapat menavigasi ke [X.Y.Z](#) tautan untuk melihat versi terbaru yang tersedia.)

```
// 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.
}
```

Perhatikan bahwa Anda tidak memerlukan plugin generator skema atau paket anotasi.

Pemetaan antara kelas Kotlin dan item DynamoDB memerlukan [ItemSchema<T>](#) implementasi, di mana T jenis kelas Kotlin. Skema terdiri dari elemen-elemen berikut:

- Konverter item, yang menentukan cara mengonversi antara instance objek Kotlin dan item DynamoDB.
- Sebuah spesifikasi kunci partisi, yang mendefinisikan nama dan jenis atribut kunci partisi.
- Secara opsional, spesifikasi kunci sortir, yang mendefinisikan nama dan jenis atribut kunci sortir.

Dalam kode berikut kita secara manual membuat sebuah `CarSchema` instance:

```
import aws.sdk.kotlin.h11.dynamodbmapper.items.ItemConverter
import aws.sdk.kotlin.h11.dynamodbmapper.items.ItemSchema
import aws.sdk.kotlin.h11.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)

```

Kode sebelumnya membuat konverter bernama `carConverter`, yang didefinisikan sebagai implementasi anonim dari `ItemConverter<Car>`. `convertTo` metode konverter menerima `Car` argumen dan mengembalikan `Item` instance yang mewakili kunci literal dan nilai atribut item DynamoDB. `convertFrom` metode konverter menerima `Item` argumen dan mengembalikan `Car` instance dari nilai atribut `Item` argumen.

Selanjutnya kode membuat dua spesifikasi utama: satu untuk kunci partisi dan satu untuk kunci sortir. Setiap tabel atau indeks DynamoDB harus memiliki persis satu kunci partisi dan, dengan demikian, demikian juga setiap definisi skema DynamoDB Mapper. Skema mungkin juga memiliki satu kunci pengurutan.

Dalam pernyataan terakhir, kode membuat skema untuk tabel `cars` DynamoDB dari konverter dan spesifikasi utama.

Skema yang dihasilkan setara dengan skema berbasis anotasi yang kami hasilkan di bagian tersebut. [the section called “Tentukan skema dengan anotasi kelas”](#) Untuk referensi, berikut ini adalah kelas beranotasi yang kami gunakan:

Kelas mobil dengan anotasi DynamoDB Mapper

```

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

    @DynamoDbSortKey
    val model: String,

    val initialYear: Int

```




```
)
```

Selain mengimplementasikan milik Anda sendiri `ItemConverter`, `DynamoDB Mapper` menyertakan beberapa implementasi bermanfaat seperti:

- [SimpleItemConverter](#): menyediakan logika konversi sederhana dengan menggunakan kelas pembangun dan deskripsi atribut. Lihat contoh di [the section called “Tentukan konverter item khusus”](#) untuk bagaimana Anda dapat menggunakan implementasi ini.
- [HeterogeneousItemConverter](#): menyediakan logika konversi tipe polimorfik dengan menggunakan atribut diskriminator dan mendelegasikan `ItemConverter` instance untuk subtype.
- [DocumentConverter](#): menyediakan logika konversi untuk data tidak terstruktur dalam `Document` objek.

## Gunakan indeks sekunder dengan `DynamoDB Mapper`

 `DynamoDB Mapper` adalah rilis Pratinjau Pengembang. Ini tidak lengkap fitur dan dapat berubah.

### Tentukan skema untuk indeks sekunder

Tabel `DynamoDB` mendukung indeks sekunder yang menyediakan akses ke data menggunakan kunci yang berbeda dari yang didefinisikan pada tabel dasar itu sendiri. Seperti tabel dasar, `DynamoDB Mapper` berinteraksi dengan indeks dengan menggunakan tipe. [ItemSchema](#)

Indeks sekunder `DynamoDB` tidak diperlukan untuk memuat setiap atribut dari tabel dasar. Dengan demikian, kelas Kotlin yang memetakan ke indeks mungkin berbeda dari kelas Kotlin yang memetakan ke tabel dasar indeks tersebut. Ketika itu terjadi, skema terpisah harus dideklarasikan untuk kelas indeks.

Kode berikut secara manual membuat skema indeks untuk tabel `DynamoDB cars`.

```
import aws.sdk.kotlin.h11.dynamodbmapper.items.ItemConverter
import aws.sdk.kotlin.h11.dynamodbmapper.items.ItemSchema
import aws.sdk.kotlin.h11.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
        )
*/

```

Kita sekarang dapat menggunakan `Model` instance dalam operasi.

Gunakan indeks sekunder dalam operasi

DynamoDB Mapper mendukung subset operasi pada indeks, yaitu `queryPaginated` dan `scanPaginated`. Untuk menjalankan operasi ini pada indeks, Anda harus terlebih dahulu


mendapatkan referensi ke indeks dari objek tabel. Dalam contoh berikut, kami menggunakan `modelSchema` yang kami buat sebelumnya untuk `cars-by-model` indeks (pembuatan tidak ditampilkan di sini):

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

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

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

## Gunakan ekspresi

 DynamoDB Mapper adalah rilis Pratinjau Pengembang. Ini tidak lengkap fitur dan dapat berubah.

Operasi DynamoDB tertentu [menerima](#) ekspresi yang dapat Anda gunakan untuk menentukan batasan atau kondisi. DynamoDB Mapper menyediakan Kotlin idiomatik DSL untuk membuat ekspresi. DSL Membawa struktur dan keterbacaan yang lebih besar ke kode Anda dan juga membuatnya lebih mudah untuk menulis ekspresi.

Bagian ini menjelaskan DSL sintaks dan memberikan berbagai contoh.

### Gunakan ekspresi dalam operasi

Anda menggunakan ekspresi dalam operasi seperti `scan`, di mana mereka memfilter item yang dikembalikan berdasarkan kriteria yang Anda tentukan. Untuk menggunakan ekspresi dengan DynamoDB Mapper, tambahkan komponen ekspresi dalam permintaan operasi.

Cuplikan berikut menunjukkan contoh ekspresi filter yang digunakan dalam operasi `scan`. Ini menggunakan argumen lambda untuk menggambarkan kriteria filter yang membatasi item yang akan dikembalikan ke item dengan nilai `year` atribut 2001:

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

table.scanPaginated {
```

```

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

```

Contoh berikut menunjukkan query operasi yang mendukung ekspresi di dua tempat—sort key filtering dan non-key filtering:

```

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

```

Filter kode sebelumnya menghasilkan yang memenuhi ketiga kriteria:

- Nilai atribut kunci partisi adalah 1000 - AND -
- Urutkan nilai atribut kunci dimulai dengan huruf M - AND -
- nilai atribut tahun adalah 2001

## Komponen-komponen DSL

DSL sintaks mengekspos beberapa jenis komponen—dijelaskan di bawah ini—yang Anda gunakan untuk membangun ekspresi.

### Atribut

Sebagian besar kondisi referensi atribut, yang diidentifikasi oleh kunci atau jalur dokumen mereka. Dengan DSK, Anda membuat semua referensi atribut dengan menggunakan `attr` fungsi dan secara opsional membuat modifikasi tambahan.

Kode berikut menunjukkan rentang referensi atribut contoh dari yang sederhana hingga yang kompleks, seperti daftar dereferensi berdasarkan indeks dan dereferensi peta dengan kunci::

```

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`.
```

## Kesetaraan dan ketidaksetaraan

Anda dapat membandingkan nilai atribut dalam ekspresi dengan persamaan dan ketidaksetaraan. Anda dapat membandingkan nilai atribut dengan nilai literal atau nilai atribut lainnya. Fungsi yang Anda gunakan untuk menentukan kondisi adalah:

- `eq`: sama dengan (setara dengan`==`)
- `neq`: tidak sama dengan (setara dengan`!=`)
- `gt`: lebih besar dari (setara dengan`>`)
- `gte`: lebih besar dari atau sama dengan (setara dengan`>=`)
- `lt`: kurang dari (setara dengan`<`)
- `lte`: kurang dari atau sama dengan (setara dengan`<=`)

Anda menggabungkan fungsi perbandingan dengan argumen dengan menggunakan notasi infix seperti yang ditunjukkan pada contoh berikut:

```
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`.
```

## Rentang dan set

Selain nilai tunggal, Anda dapat membandingkan nilai atribut dengan beberapa nilai dalam rentang atau set. Anda menggunakan [isIn](#) fungsi infix untuk melakukan perbandingan seperti yang ditunjukkan pada contoh berikut:

```
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` fungsi ini menyediakan kelebihan beban untuk koleksi (seperti `Set<String>`) dan batas yang dapat Anda ekspresikan sebagai Kotlin `ClosedRange<T>` (seperti). `IntRange` Untuk batas yang tidak dapat Anda ekspresikan sebagai `ClosedRange<T>` (seperti array byte atau referensi atribut lainnya), Anda dapat menggunakan fungsi: `isBetween`

```

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`.

```

## Logika Boolean

Anda dapat menggabungkan kondisi individual atau diubah menggunakan logika boolean dengan menggunakan fungsi-fungsi berikut:

- `and`: setiap kondisi harus benar (setara dengan `&&`)
- `or`: setidaknya satu kondisi harus benar (setara dengan `| |`)
- `not`: kondisi yang diberikan harus salah (setara dengan `!`)

Contoh berikut menunjukkan setiap fungsi:

```

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.
)

```

Anda selanjutnya dapat menggabungkan kondisi boolean dengan fungsi boolean untuk membuat logika bersarang seperti yang ditunjukkan pada ekspresi berikut:

```

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

```

Filter ekspresi sebelumnya menghasilkan filter yang memenuhi salah satu dari kondisi ini:

- Kedua kondisi ini benar:
  - foo nilai atribut adalah 123 - AND -
  - bar nilai atribut adalah "abc"
- Kedua kondisi ini benar:
  - foo nilai atribut adalah 234 - - AND
  - bar nilai atribut adalah "bcd"

Ini setara dengan ekspresi boolean Kotlin berikut:

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

## Fungsi dan properti

Fungsi dan properti berikut memberikan kemampuan ekspresi tambahan:

- [contains](#): memeriksa apakah nilai atribut string/list berisi nilai yang diberikan
- [exists](#): memeriksa apakah atribut didefinisikan dan menyimpan nilai apa pun (termasuk null)

- [notExists](#): memeriksa apakah atribut tidak terdefinisi
- [isOfType](#): memeriksa apakah nilai atribut dari jenis tertentu, seperti string, nomor, boolean, dan sebagainya
- [size](#): mendapatkan ukuran atribut, seperti jumlah elemen dalam koleksi atau panjang string
- [startsWith](#): memeriksa apakah nilai atribut string dimulai dengan substring yang diberikan

Contoh berikut menunjukkan penggunaan fungsi dan properti tambahan yang dapat Anda gunakan dalam ekspresi:

```
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.
```

## Urutkan filter kunci

Ekspresi filter pada kunci sortir (seperti dalam `keyCondition` parameter `query` operasi) tidak menggunakan nilai atribut bernama. Untuk menggunakan kunci sortir dalam filter, Anda harus menggunakan kata kunci `sortKey` dalam semua perbandingan. `sortKeyKata` kunci menggantikan `attr("<sort key name>")` seperti yang ditunjukkan dalam contoh berikut:

```
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).
```

Anda tidak dapat menggabungkan filter kunci sortir dengan logika boolean dan mereka hanya mendukung sebagian dari perbandingan yang dijelaskan di atas:

- [Persamaan dan ketidaksetaraan: semua perbandingan](#) didukung



- [Rentang dan set](#): semua perbandingan didukung
- [Logika Boolean](#): tidak didukung
- [Fungsi dan properti](#): `startsWith` hanya didukung

# Contoh kode SDK untuk Kotlin

Contoh kode dalam topik ini menunjukkan cara menggunakan AWS SDK untuk Kotlin. AWS

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Beberapa layanan berisi kategori contoh tambahan yang menunjukkan cara memanfaatkan pustaka atau fungsi khusus untuk layanan.

## Layanan

- [Contoh API Gateway menggunakan SDK untuk Kotlin](#)
- [Contoh Aurora menggunakan SDK untuk Kotlin](#)
- [Contoh Auto Scaling menggunakan SDK untuk Kotlin](#)
- [Contoh Amazon Bedrock menggunakan SDK untuk Kotlin](#)
- [CloudWatch contoh menggunakan SDK untuk Kotlin](#)
- [CloudWatch Contoh log menggunakan SDK untuk Kotlin](#)
- [Contoh Penyedia Identitas Amazon Cognito menggunakan SDK untuk Kotlin](#)
- [Amazon Comprehend contoh menggunakan SDK untuk Kotlin](#)
- [Contoh DynamoDB menggunakan SDK untuk Kotlin](#)
- [EC2 Contoh Amazon menggunakan SDK untuk Kotlin](#)
- [Contoh Amazon ECR menggunakan SDK untuk Kotlin](#)
- [OpenSearch Contoh layanan menggunakan SDK untuk Kotlin](#)
- [EventBridge contoh menggunakan SDK untuk Kotlin](#)
- [AWS Glue contoh menggunakan SDK untuk Kotlin](#)
- [Contoh IAM menggunakan SDK untuk Kotlin](#)
- [AWS IoT contoh menggunakan SDK untuk Kotlin](#)

- [AWS IoT data contoh menggunakan SDK untuk Kotlin](#)
- [Contoh Amazon Keyspaces menggunakan SDK untuk Kotlin](#)
- [AWS KMS contoh menggunakan SDK untuk Kotlin](#)
- [Contoh Lambda menggunakan SDK untuk Kotlin](#)
- [MediaConvert contoh menggunakan SDK untuk Kotlin](#)
- [Amazon Pinpoint contoh menggunakan SDK untuk Kotlin](#)
- [Contoh Amazon RDS menggunakan SDK untuk Kotlin](#)
- [Contoh Amazon RDS Data Service menggunakan SDK untuk Kotlin](#)
- [Contoh Amazon Redshift menggunakan SDK untuk Kotlin](#)
- [Contoh Rekognition Amazon menggunakan SDK untuk Kotlin](#)
- [Route 53 contoh pendaftaran domain menggunakan SDK untuk Kotlin](#)
- [Contoh Amazon S3 menggunakan SDK untuk Kotlin](#)
- [SageMaker Contoh AI menggunakan SDK untuk Kotlin](#)
- [Contoh Secrets Manager menggunakan SDK untuk Kotlin](#)
- [Contoh Amazon SES menggunakan SDK untuk Kotlin](#)
- [Contoh Amazon SNS menggunakan SDK untuk Kotlin](#)
- [Contoh Amazon SQS menggunakan SDK untuk Kotlin](#)
- [Contoh Step Functions menggunakan SDK untuk Kotlin](#)
- [Dukungan contoh menggunakan SDK untuk Kotlin](#)
- [Contoh Amazon Translate menggunakan SDK untuk Kotlin](#)

## Contoh API Gateway menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan API Gateway.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Skenario](#)

## Skenario

Membuat aplikasi nirserver untuk mengelola foto

Contoh kode berikut menunjukkan cara membuat aplikasi tanpa server yang memungkinkan pengguna mengelola foto menggunakan label.

SDK untuk Kotlin

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

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

## Contoh Aurora menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Aurora.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Hal-hal mendasar](#)
- [Tindakan](#)
- [Skenario](#)

## Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara:

- Membuat grup parameter klaster DB Aurora dan mengatur nilai parameter.
- Membuat klaster DB yang menggunakan grup parameter.
- Membuat instans DB yang berisi basis data.
- Mengambil snapshot klaster DB, lalu membersihkan sumber daya.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
/**
```

```
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](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(InterruptedExcetion::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(s1Time * 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 { engine0b ->
        println("The name of the DB parameter group family for the database
engine is ${engine0b.dbParameterGroupFamily}")
        println("The name of the database engine ${engine0b.engine}")
        println("The version number of the database engine
${engine0b.engineVersion}")
    }
}
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [Buat DBCluster](#)
  - [Buat DBCluster ParameterGroup](#)
  - [Buat DBCluster Snapshot](#)
  - [Buat DBInstance](#)
  - [Hapus DBCluster](#)
  - [Hapus DBCluster ParameterGroup](#)
  - [Hapus DBInstance](#)
  - [Jelaskan DBCluster ParameterGroups](#)
  - [Jelaskan DBCluster Parameter](#)
  - [Jelaskan DBCluster Snapshots](#)
  - [Jelaskan DBClusters](#)
  - [Jelaskan DBEngine Versi](#)
  - [Jelaskan DBInstances](#)
  - [DescribeOrderableDBInstancePilihan](#)
  - [Memodifikasi DBCluster ParameterGroup](#)

# Tindakan

## CreateDBCluster

Contoh kode berikut menunjukkan cara menggunakan `CreateDBCluster`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}
```

- Untuk detail API, lihat [Membuat DBCluster](#) di AWS SDK untuk referensi API Kotlin.

## CreateDBClusterParameterGroup

Contoh kode berikut menunjukkan cara menggunakan `CreateDBClusterParameterGroup`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [Membuat DBCluster ParameterGroup](#) di AWS SDK untuk referensi API Kotlin.

## CreateDBClusterSnapshot

Contoh kode berikut menunjukkan cara menggunakan `CreateDBClusterSnapshot`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [Membuat DBCluster Snapshot](#) di AWS SDK untuk referensi API Kotlin.

## CreateDBInstance

Contoh kode berikut menunjukkan cara menggunakan CreateDBInstance.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}
```

- Untuk detail API, lihat [Membuat DBInstance](#) di AWS SDK untuk referensi API Kotlin.

## DeleteDBCluster

Contoh kode berikut menunjukkan cara menggunakan DeleteDBCluster.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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!")
}
}

```

- Untuk detail API, lihat [Menghapus DBCluster](#) di AWS SDK untuk referensi API Kotlin.

## DeleteDBClusterParameterGroup

Contoh kode berikut menunjukkan cara menggunakan `DeleteDBClusterParameterGroup`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

@Throws(InterruptedExcption::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.")
}
}

```

- Untuk detail API, lihat [Menghapus DBCluster ParameterGroup](#) di AWS SDK untuk referensi API Kotlin.

## DeleteDBInstance

Contoh kode berikut menunjukkan cara menggunakan `DeleteDBInstance`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [Menghapus DBInstance](#) di AWS SDK untuk referensi API Kotlin.

## DescribeDBClusterParameterGroups

Contoh kode berikut menunjukkan cara menggunakan `DescribeDBClusterParameterGroups`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
}

```

- Untuk detail API, lihat [Menjelaskan DBCluster ParameterGroups](#) di AWS SDK untuk referensi API Kotlin.

## DescribeDBClusterParameters

Contoh kode berikut menunjukkan cara menggunakan `DescribeDBClusterParameters`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repository Contoh Kode AWS](#).

```

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}")
                }
            }
        }
    }
}

```

- Untuk detail API, lihat [Menjelaskan DBCluster Parameter](#) di AWS SDK untuk referensi API Kotlin.

## DescribeDBClusterSnapshots

Contoh kode berikut menunjukkan cara menggunakan `DescribeDBClusterSnapshots`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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(1Time * 5000)
                }
            }
        }
    }
}
println("The Snapshot is available!")
}
```

- Untuk detail API, lihat [Menjelaskan DBCluster Snapshot](#) di AWS SDK untuk referensi API Kotlin.

## DescribeDBClusters

Contoh kode berikut menunjukkan cara menggunakan `DescribeDBClusters`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repository Contoh Kode AWS](#).

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

```

    }
  }
}

```

- Untuk detail API, lihat [Menjelaskan DBClusters](#) di AWS SDK untuk referensi API Kotlin.

## DescribeDBEngineVersions

Contoh kode berikut menunjukkan cara menggunakan `DescribeDBEngineVersions`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

// 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}")
        }
    }
}

```

- Untuk detail API, lihat [Menjelaskan DBEngine Versi](#) di AWS SDK untuk referensi API Kotlin.

## DescribeDBInstances

Contoh kode berikut menunjukkan cara menggunakan `DescribeDBInstances`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
}
```

- Untuk detail API, lihat [Menjelaskan DBInstances](#) di AWS SDK untuk referensi API Kotlin.

## ModifyDBClusterParameterGroup

Contoh kode berikut menunjukkan cara menggunakan `ModifyDBClusterParameterGroup`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [Memodifikasi DBCluster ParameterGroup](#) di AWS SDK untuk referensi API Kotlin.

## Skenario

Buat pelacak butir kerja Aurora Nirserver

Contoh kode berikut menunjukkan cara membuat aplikasi web yang melacak item pekerjaan dalam database Amazon Aurora Tanpa Server dan menggunakan Amazon Simple Email Service (Amazon SES) untuk mengirim laporan.

SDK untuk Kotlin

Menunjukkan cara membuat aplikasi web yang melacak dan melaporkan butir kerja yang tersimpan dalam basis data Amazon RDS.

Untuk kode sumber lengkap dan petunjuk tentang cara menyiapkan Spring REST API yang menanyakan data Amazon Aurora Tanpa Server dan untuk digunakan oleh aplikasi React, lihat contoh lengkapnya di [GitHub](#)

Layanan yang digunakan dalam contoh ini

- Aurora
- Amazon RDS
- Layanan Data Amazon RDS
- Amazon SES

## Contoh Auto Scaling menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Auto Scaling.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik



- [Hal-hal mendasar](#)
- [Tindakan](#)

## Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara:

- Buat grup EC2 Auto Scaling Amazon dengan template peluncuran dan Availability Zone, dan dapatkan informasi tentang menjalankan instans.
- Aktifkan pengumpulan CloudWatch metrik Amazon.
- Perbarui kapasitas yang diinginkan grup dan tunggu instance dimulai.
- Mengakhiri sebuah instance dalam grup.
- Buat daftar aktivitas penskalaan yang terjadi sebagai respons terhadap permintaan pengguna dan perubahan kapasitas.
- Dapatkan statistik untuk CloudWatch metrik, lalu bersihkan sumber daya.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.numberOfWorkAutoScalingGroups}")
        }
    }

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")
    }
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [CreateAutoScalingGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DescribeAutoScalingGroups](#)
  - [DescribeAutoScalingInstances](#)



- [DescribeScalingActivities](#)
- [DisableMetricsCollection](#)
- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

## Tindakan

### CreateAutoScalingGroup

Contoh kode berikut menunjukkan cara menggunakan `CreateAutoScalingGroup`.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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!")
    }
}

```

- Untuk detail API, lihat [CreateAutoScalingGroup](#) di AWS SDK untuk referensi API Kotlin.

## DeleteAutoScalingGroup

Contoh kode berikut menunjukkan cara menggunakan `DeleteAutoScalingGroup`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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")
    }
}

```

- Untuk detail API, lihat [DeleteAutoScalingGroup](#) di AWS SDK untuk referensi API Kotlin.

## DescribeAutoScalingGroups

Contoh kode berikut menunjukkan cara menggunakan `DescribeAutoScalingGroups`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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)
                }
            }
        }
    }
}

```

- Untuk detail API, lihat [DescribeAutoScalingGroups](#) di AWS SDK untuk referensi API Kotlin.

## DescribeAutoScalingInstances

Contoh kode berikut menunjukkan cara menggunakan `DescribeAutoScalingInstances`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [DescribeAutoScalingInstances](#) di AWS SDK untuk referensi API Kotlin.

## DescribeScalingActivities

Contoh kode berikut menunjukkan cara menggunakan `DescribeScalingActivities`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [DescribeScalingActivities](#) di AWS SDK untuk referensi API Kotlin.

## DisableMetricsCollection

Contoh kode berikut menunjukkan cara menggunakan `DisableMetricsCollection`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

```
}  
}
```

- Untuk detail API, lihat [DisableMetricsCollection](#) di AWS SDK untuk referensi API Kotlin.

## EnableMetricsCollection

Contoh kode berikut menunjukkan cara menggunakan `EnableMetricsCollection`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")  
    }  
}
```

- Untuk detail API, lihat [EnableMetricsCollection](#) di AWS SDK untuk referensi API Kotlin.

## SetDesiredCapacity

Contoh kode berikut menunjukkan cara menggunakan `SetDesiredCapacity`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [SetDesiredCapacity](#) di AWS SDK untuk referensi API Kotlin.

## TerminateInstanceInAutoScalingGroup

Contoh kode berikut menunjukkan cara menggunakan `TerminateInstanceInAutoScalingGroup`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}

```

- Untuk detail API, lihat [TerminateInstanceInAutoScalingGroup](#) di AWS SDK untuk referensi API Kotlin.

## UpdateAutoScalingGroup

Contoh kode berikut menunjukkan cara menggunakan `UpdateAutoScalingGroup`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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")
    }
}
```

- Untuk detail API, lihat [UpdateAutoScalingGroup](#) di AWS SDK untuk referensi API Kotlin.

## Contoh Amazon Bedrock menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Amazon Bedrock.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Tindakan](#)

## Tindakan

### ListFoundationModels

Contoh kode berikut menunjukkan cara menggunakan ListFoundationModels.

## SDK untuk Kotlin

**Note**

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat daftar model foundation Amazon Bedrock yang tersedia.

```
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
        }
    }
}
```

- Untuk detail API, lihat [ListFoundationModels](#) di AWS SDK untuk referensi API Kotlin.

## CloudWatch contoh menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin with. CloudWatch

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.


Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Memulai

Halo CloudWatch

Contoh kode berikut menunjukkan cara untuk mulai menggunakan CloudWatch.

SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListMetrics](#) di AWS SDK untuk referensi API Kotlin.

## Topik

- [Hal-hal mendasar](#)
- [Tindakan](#)

## Hal-hal mendasar

### Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara:

- Buat daftar CloudWatch ruang nama dan metrik.
- Ambil statistik untuk metrik dan estimasi penagihan.
- Membuat dan memperbarui sebuah dasbor.
- Membuat dan menambahkan data ke metrik.
- Membuat dan memicu alarm, lalu lihat riwayat alarm.

- Menambahkan detektor anomali.
- Ambil gambar metrik, lalu bersihkan sumber daya.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

Jalankan skenario interaktif yang menunjukkan CloudWatch fitur.

```
/**
```

```
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
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [DeleteAlarms](#)
  - [DeleteAnomalyDetector](#)
  - [DeleteDashboards](#)
  - [DescribeAlarmHistory](#)
  - [DescribeAlarms](#)
  - [DescribeAlarmsForMetric](#)
  - [DescribeAnomalyDetectors](#)
  - [GetMetricData](#)
  - [GetMetricStatistics](#)
  - [GetMetricWidgetImage](#)
  - [ListMetrics](#)
  - [PutAnomalyDetector](#)
  - [PutDashboard](#)
  - [PutMetricAlarm](#)
  - [PutMetricData](#)

## Tindakan

### DeleteAlarms

Contoh kode berikut menunjukkan cara menggunakan `DeleteAlarms`.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
}
}

```

- Untuk detail API, lihat [DeleteAlarms](#) di AWS SDK untuk referensi API Kotlin.

## DeleteAnomalyDetector

Contoh kode berikut menunjukkan cara menggunakan `DeleteAnomalyDetector`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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.")
    }
}
```

- Untuk detail API, lihat [DeleteAnomalyDetector](#) di AWS SDK untuk referensi API Kotlin.

## DeleteDashboards

Contoh kode berikut menunjukkan cara menggunakan `DeleteDashboards`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.")
    }
}
```

- Untuk detail API, lihat [DeleteDashboards](#) di AWS SDK untuk referensi API Kotlin.

## DescribeAlarmHistory

Contoh kode berikut menunjukkan cara menggunakan `DescribeAlarmHistory`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

```

- Untuk detail API, lihat [DescribeAlarmHistory](#) di AWS SDK untuk referensi API Kotlin.

## DescribeAlarms

Contoh kode berikut menunjukkan cara menggunakan `DescribeAlarms`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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}")
        }
    }
}
}

```

- Untuk detail API, lihat [DescribeAlarms](#) di AWS SDK untuk referensi API Kotlin.

## DescribeAlarmsForMetric

Contoh kode berikut menunjukkan cara menggunakan `DescribeAlarmsForMetric`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

```

    }
}
}

```

- Untuk detail API, lihat [DescribeAlarmsForMetric](#) di AWS SDK untuk referensi API Kotlin.

## DescribeAnomalyDetectors

Contoh kode berikut menunjukkan cara menggunakan `DescribeAnomalyDetectors`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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}")
        }
    }
}
}

```

- Untuk detail API, lihat [DescribeAnomalyDetectors](#) di AWS SDK untuk referensi API Kotlin.

## DisableAlarmActions

Contoh kode berikut menunjukkan cara menggunakan `DisableAlarmActions`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [DisableAlarmActions](#) di AWS SDK untuk referensi API Kotlin.

## EnableAlarmActions

Contoh kode berikut menunjukkan cara menggunakan `EnableAlarmActions`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [EnableAlarmActions](#) di AWS SDK untuk referensi API Kotlin.

## GetMetricData

Contoh kode berikut menunjukkan cara menggunakan `GetMetricData`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
        }
    }
```

```
}  
}
```

- Untuk detail API, lihat [GetMetricData](#) di AWS SDK untuk referensi API Kotlin.

## GetMetricStatistics

Contoh kode berikut menunjukkan cara menggunakan `GetMetricStatistics`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
        }
    }
}
}

```

- Untuk detail API, lihat [GetMetricStatistics](#) di AWS SDK untuk referensi API Kotlin.

## GetMetricWidgetImage

Contoh kode berikut menunjukkan cara menggunakan `GetMetricWidgetImage`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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")
}

```

- Untuk detail API, lihat [GetMetricWidgetImage](#) di AWS SDK untuk referensi API Kotlin.

## ListDashboards

Contoh kode berikut menunjukkan cara menggunakan `ListDashboards`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
        }
    }
}

```

- Untuk detail API, lihat [ListDashboards](#) di AWS SDK untuk referensi API Kotlin.

## ListMetrics

Contoh kode berikut menunjukkan cara menggunakan `ListMetrics`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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
}

```

- Untuk detail API, lihat [ListMetrics](#) di AWS SDK untuk referensi API Kotlin.

## PutAnomalyDetector

Contoh kode berikut menunjukkan cara menggunakan PutAnomalyDetector.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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.")
    }
}

```

```
}
```

- Untuk detail API, lihat [PutAnomalyDetector](#) di AWS SDK untuk referensi API Kotlin.

## PutDashboard

Contoh kode berikut menunjukkan cara menggunakan `PutDashboard`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

```
}
```

- Untuk detail API, lihat [PutDashboard](#) di AWS SDK untuk referensi API Kotlin.

## PutMetricAlarm

Contoh kode berikut menunjukkan cara menggunakan `PutMetricAlarm`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun putMetricAlarm(
    alarmNameVal: String,
    instanceIdVal: String,
) {
    val dimension0b =
        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(dimension0b)
```



```

    }

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

```

- Untuk detail API, lihat [PutMetricAlarm](#) di AWS SDK untuk referensi API Kotlin.

## PutMetricData

Contoh kode berikut menunjukkan cara menggunakan `PutMetricData`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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")
    }
}

```

- Untuk detail API, lihat [PutMetricData](#) di AWS SDK untuk referensi API Kotlin.

## CloudWatch Contoh log menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan CloudWatch Log.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

## Topik

- [Tindakan](#)

## Tindakan

### DeleteSubscriptionFilter

Contoh kode berikut menunjukkan cara menggunakan DeleteSubscriptionFilter.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [DeleteSubscriptionFilter](#) di AWS SDK untuk referensi API Kotlin.

### DescribeSubscriptionFilters

Contoh kode berikut menunjukkan cara menggunakan DescribeSubscriptionFilters.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [DescribeSubscriptionFilters](#) di AWS SDK untuk referensi API Kotlin.

## StartLiveTail

Contoh kode berikut menunjukkan cara menggunakan `StartLiveTail`.

## SDK untuk Kotlin

Sertakan file-file yang diperlukan.

```
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
```

## Mulai sesi Live Tail.

```
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 unsubscribe 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)
}
```

- Untuk detail API, lihat [StartLiveTail](#) di AWS SDK untuk referensi API Kotlin.

# Contoh Penyedia Identitas Amazon Cognito menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Penyedia Identitas Amazon Cognito.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Tindakan](#)
- [Skenario](#)

## Tindakan

### AdminGetUser

Contoh kode berikut menunjukkan cara menggunakan `AdminGetUser`.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
}
}

```

- Untuk detail API, lihat [AdminGetUser](#) di AWS SDK untuk referensi API Kotlin.

## AdminInitiateAuth

Contoh kode berikut menunjukkan cara menggunakan `AdminInitiateAuth`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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
}
}

```

- Untuk detail API, lihat [AdminInitiateAuth](#) di AWS SDK untuk referensi API Kotlin.

## AdminRespondToAuthChallenge

Contoh kode berikut menunjukkan cara menggunakan `AdminRespondToAuthChallenge`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponses0b = mutableMapOf<String, String>()
    challengeResponses0b["USERNAME"] = userName
    challengeResponses0b["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {

```



```

        challengeName = ChallengeNameType.SoftwareTokenMfa
        clientId = clientIdVal
        challengeResponses = challengeResponses0b
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
{ identityProviderClient ->
    val respondToAuthChallengeResult =
identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
    println("respondToAuthChallengeResult.getAuthenticationResult()
${respondToAuthChallengeResult.authenticationResult}")
}
}

```

- Untuk detail API, lihat [AdminRespondToAuthChallenge](#) di AWS SDK untuk referensi API Kotlin.

## AssociateSoftwareToken

Contoh kode berikut menunjukkan cara menggunakan `AssociateSoftwareToken`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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
    }
}
```

- Untuk detail API, lihat [AssociateSoftwareToken](#) di AWS SDK untuk referensi API Kotlin.

## ConfirmSignUp

Contoh kode berikut menunjukkan cara menggunakan `ConfirmSignUp`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [ConfirmSignUp](#) di AWS SDK untuk referensi API Kotlin.

## ListUsers

Contoh kode berikut menunjukkan cara menggunakan `ListUsers`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListUsers](#) di AWS SDK untuk referensi API Kotlin.

## ResendConfirmationCode

Contoh kode berikut menunjukkan cara menggunakan `ResendConfirmationCode`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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))
    }
}
```

- Untuk detail API, lihat [ResendConfirmationCode](#) di AWS SDK untuk referensi API Kotlin.

## SignUp

Contoh kode berikut menunjukkan cara menggunakan SignUp.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}

```

- Untuk detail API, lihat [SignUp](#) di AWS SDK untuk referensi API Kotlin.

## VerifySoftwareToken

Contoh kode berikut menunjukkan cara menggunakan `VerifySoftwareToken`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

// 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}")
}
}
```

- Untuk detail API, lihat [VerifySoftwareToken](#) di AWS SDK untuk referensi API Kotlin.

## Skenario

Mendaftar pengguna dengan kumpulan pengguna yang membutuhkan MFA

Contoh kode berikut ini menunjukkan cara:

- Daftar dan konfirmasi pengguna dengan nama pengguna, kata sandi, dan alamat email.
- Siapkan otentikasi multi-faktor dengan mengaitkan aplikasi MFA dengan pengguna.
- Masuk dengan menggunakan kata sandi dan kode MFA.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
/**
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 in0b = Scanner(System.`in`)
```

```
val userName = in0b.nextLine()
println(userName)

println("**** Enter your password")
val password: String = in0b.nextLine()

println("**** Enter your email")
val email = in0b.nextLine()

println("**** Signing up $userName")
signUp(clientId, userName, password, email)

println("**** Getting $userName in the user pool")
getAdminUser(userName, poolId)

println("**** Confirmation code sent to $userName. Would you like to send a new
code? (Yes/No)")
val ans = in0b.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 = in0b.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 = in0b.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 = in0b.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")
    }
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [AdminGetUser](#)
  - [AdminInitiateAuth](#)
  - [AdminRespondToAuthChallenge](#)
  - [AssociateSoftwareToken](#)
  - [ConfirmDevice](#)
  - [ConfirmSignUp](#)
  - [InitiateAuth](#)
  - [ListUsers](#)
  - [ResendConfirmationCode](#)
  - [RespondToAuthChallenge](#)
  - [SignUp](#)
  - [VerifySoftwareToken](#)

## Amazon Comprehend contoh menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Amazon Comprehend.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Skenario](#)

## Skenario

Buat aplikasi perpesanan

Contoh kode berikut menunjukkan cara membuat aplikasi pesan dengan menggunakan Amazon SQS.

SDK untuk Kotlin

Menunjukkan cara menggunakan Amazon SQS API untuk mengembangkan Spring REST API yang mengirim dan mengambil pesan.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Comprehend
- Amazon SQS

## Contoh DynamoDB menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan DynamoDB.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Hal-hal mendasar](#)

- [Tindakan](#)
- [Skenario](#)

## Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara:

- Buat tabel yang dapat menyimpan data film.
- Masukkan, dapatkan, dan perbarui satu film dalam tabel tersebut.
- Tulis data film ke tabel dari file JSON sampel.
- Kueri untuk film yang dirilis pada tahun tertentu.
- Pindai film yang dirilis dalam suatu rentang tahun.
- Hapus film dari tabel, lalu hapus tabel tersebut.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat tabel DynamoDB.

```
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}")
    }
}
```



Buat fungsi pembantu untuk mengunduh dan mengekstrak file JSON sampel.

```
// 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.")
    }
}

```

Dapatkan item dari tabel.

```

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)
        }
    }
}

```

Contoh lengkap.

```

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.")
```

```
deleteIssuesTable(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 {
```

```
            keyType = KeyType.H
```

```
            keyName = key
```

```
        }
```

```
    }
```

```
        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]}")
            }
        }
    }
}

```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk Kotlin.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)

- [DescribeTable](#)
- [GetItem](#)
- [PutItem](#)
- [Kueri](#)
- [Scan](#)
- [UpdateItem](#)

## Tindakan

### CreateTable

Contoh kode berikut menunjukkan cara menggunakan CreateTable.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}

```

- Untuk detail API, lihat [CreateTable](#) di AWS SDK untuk referensi API Kotlin.

## DeleteItem

Contoh kode berikut menunjukkan cara menggunakan `DeleteItem`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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")
        }
    }
}
```

- Untuk detail API, lihat [DeleteItem](#) di AWS SDK untuk referensi API Kotlin.

## DeleteTable

Contoh kode berikut menunjukkan cara menggunakan `DeleteTable`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun deleteDynamoDBTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

```
    }  
}
```

- Untuk detail API, lihat [DeleteTable](#) di AWS SDK untuk referensi API Kotlin.

## GetItem

Contoh kode berikut menunjukkan cara menggunakan `GetItem`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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)  
        }  
    }  
}
```

- Untuk detail API, lihat [GetItem](#) di AWS SDK untuk referensi API Kotlin.

## ListTables

Contoh kode berikut menunjukkan cara menggunakan `ListTables`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun listAllTables() {
    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.listTables(ListTablesRequest {})
        response.tableNames?.forEach { tableName ->
            println("Table name is $tableName")
        }
    }
}
```

- Untuk detail API, lihat [ListTables](#) di AWS SDK untuk referensi API Kotlin.

## PutItem

Contoh kode berikut menunjukkan cara menggunakan `PutItem`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.")
    }
}
```

- Untuk detail API, lihat [PutItem](#) di AWS SDK untuk referensi API Kotlin.

## Query

Contoh kode berikut menunjukkan cara menggunakan Query.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}
```

- Untuk detail API, lihat [Kueri](#) di Referensi API AWS SDK untuk Kotlin.

## Scan

Contoh kode berikut menunjukkan cara menggunakan Scan.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [Scan](#) di Referensi API AWS SDK untuk Kotlin.

## UpdateItem

Contoh kode berikut menunjukkan cara menggunakan UpdateItem.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [UpdateItem](#) di AWS SDK untuk referensi API Kotlin.

## Skenario

Membangun aplikasi untuk mengirimkan data ke tabel DynamoDB

Contoh kode berikut menunjukkan cara membuat aplikasi yang mengirimkan data ke tabel Amazon DynamoDB dan memberi tahu Anda saat pengguna memperbarui tabel.



## SDK untuk Kotlin

Menunjukkan cara membuat aplikasi Android native yang mengirimkan data menggunakan API Kotlin Amazon DynamoDB dan mengirim pesan teks menggunakan API Kotlin Amazon SNS.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- DynamoDB
- Amazon SNS

## Membuat aplikasi nirserver untuk mengelola foto

Contoh kode berikut menunjukkan cara membuat aplikasi tanpa server yang memungkinkan pengguna mengelola foto menggunakan label.

## SDK untuk Kotlin

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

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

## Membuat aplikasi web untuk melacak data DynamoDB

Contoh kode berikut menunjukkan cara membuat aplikasi web yang melacak item kerja dalam tabel Amazon DynamoDB dan menggunakan Amazon Simple Email Service (Amazon SES) untuk mengirim laporan.

### SDK untuk Kotlin

Menunjukkan cara menggunakan Amazon DynamoDB API untuk membuat aplikasi web dinamis yang melacak data kerja DynamoDB.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- DynamoDB
- Amazon SES

## Melakukan kueri pada tabel menggunakan batch pernyataan PartiQL

Contoh kode berikut ini menunjukkan cara:

- Dapatkan batch item dengan menjalankan beberapa pernyataan SELECT.
- Tambahkan batch item dengan menjalankan beberapa pernyataan INSERT.
- Perbarui batch item dengan menjalankan beberapa pernyataan UPDATE.
- Hapus batch item dengan menjalankan beberapa pernyataan DELETE.

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
}
}

```

- Untuk detail API, lihat [BatchExecuteStatement](#) di AWS SDK untuk referensi API Kotlin.

Melakukan kueri tabel menggunakan PartiQL

Contoh kode berikut ini menunjukkan cara:

- Dapatkan item dengan menjalankan pernyataan SELECT.
- Tambahkan item dengan menjalankan pernyataan INSERT.
- Perbarui item dengan menjalankan pernyataan UPDATE.
- Hapus item dengan menjalankan pernyataan DELETE.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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)
```

```
}
```

- Untuk detail API, lihat [ExecuteStatement](#) di AWS SDK untuk referensi API Kotlin.

## EC2 Contoh Amazon menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Amazon. EC2

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

### Memulai

#### Halo Amazon EC2

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon EC2.

#### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
}
}
```

- Untuk detail API, lihat [DescribeSecurityGroups](#) di AWS SDK untuk referensi API Kotlin.

## Topik

- [Hal-hal mendasar](#)
- [Tindakan](#)

## Hal-hal mendasar

### Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara:

- Membuat pasangan kunci dan grup keamanan.
- Memilih Amazon Machine Image (AMI) dan tipe instans yang kompatibel, lalu membuat instans.
- Menghentikan dan memulai ulang instans.
- Kaitkan alamat IP Elastis dengan instans Anda.
- Menghubungkan instans Anda dengan SSH, lalu membersihkan sumber daya.

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
/**
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.waitForInstanceTerminated {
        // 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.waitForInstanceRunning {
            // 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.waitForInstanceStopped {
            // 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")
    }
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [AllocateAddress](#)
  - [AssociateAddress](#)
  - [AuthorizeSecurityGroupIngress](#)
  - [CreateKeyPair](#)
  - [CreateSecurityGroup](#)
  - [DeleteKeyPair](#)
  - [DeleteSecurityGroup](#)
  - [DescribeImages](#)
  - [DescribeInstanceTypes](#)
  - [DescribeInstances](#)
  - [DescribeKeyPairs](#)
  - [DescribeSecurityGroups](#)
  - [DisassociateAddress](#)
  - [ReleaseAddress](#)

- [RunInstances](#)
- [StartInstances](#)
- [StopInstances](#)
- [TerminateInstances](#)
- [UnmonitorInstances](#)

## Tindakan

### AllocateAddress

Contoh kode berikut menunjukkan cara menggunakan `AllocateAddress`.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}
```

```
}
```

- Untuk detail API, lihat [AllocateAddress](#) di AWS SDK untuk referensi API Kotlin.

## AssociateAddress

Contoh kode berikut menunjukkan cara menggunakan `AssociateAddress`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}
```

- Untuk detail API, lihat [AssociateAddress](#) di AWS SDK untuk referensi API Kotlin.

## AuthorizeSecurityGroupIngress

Contoh kode berikut menunjukkan cara menggunakan `AuthorizeSecurityGroupIngress`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
}
}

```

- Untuk detail API, lihat [AuthorizeSecurityGroupIngress](#) di AWS SDK untuk referensi API Kotlin.

## CreateKeyPair

Contoh kode berikut menunjukkan cara menggunakan `CreateKeyPair`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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}")
    }
}
}

```

- Untuk detail API, lihat [CreateKeyPair](#) di AWS SDK untuk referensi API Kotlin.

## CreateSecurityGroup

Contoh kode berikut menunjukkan cara menggunakan `CreateSecurityGroup`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}

```

- Untuk detail API, lihat [CreateSecurityGroup](#) di AWS SDK untuk referensi API Kotlin.

## DeleteKeyPair

Contoh kode berikut menunjukkan cara menggunakan `DeleteKeyPair`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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")
    }
}
```

- Untuk detail API, lihat [DeleteKeyPair](#) di AWS SDK untuk referensi API Kotlin.

## DeleteSecurityGroup

Contoh kode berikut menunjukkan cara menggunakan `DeleteSecurityGroup`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [DeleteSecurityGroup](#) di AWS SDK untuk referensi API Kotlin.

## DescribeInstanceTypes

Contoh kode berikut menunjukkan cara menggunakan `DescribeInstanceTypes`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
// 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
    }
}
```

- Untuk detail API, lihat [DescribeInstanceTypes](#) di AWS SDK untuk referensi API Kotlin.

## DescribeInstances

Contoh kode berikut menunjukkan cara menggunakan `DescribeInstances`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [DescribeInstances](#) di AWS SDK untuk referensi API Kotlin.

## DescribeKeyPairs

Contoh kode berikut menunjukkan cara menggunakan `DescribeKeyPairs`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [DescribeKeyPairs](#) di AWS SDK untuk referensi API Kotlin.

## DescribeSecurityGroups

Contoh kode berikut menunjukkan cara menggunakan `DescribeSecurityGroups`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}

```

- Untuk detail API, lihat [DescribeSecurityGroups](#) di AWS SDK untuk referensi API Kotlin.

## DisassociateAddress

Contoh kode berikut menunjukkan cara menggunakan `DisassociateAddress`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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!")
    }
}

```

- Untuk detail API, lihat [DisassociateAddress](#) di AWS SDK untuk referensi API Kotlin.

## ReleaseAddress

Contoh kode berikut menunjukkan cara menggunakan `ReleaseAddress`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [ReleaseAddress](#) di AWS SDK untuk referensi API Kotlin.

## RunInstances

Contoh kode berikut menunjukkan cara menggunakan `RunInstances`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
}
}

```

- Untuk detail API, lihat [RunInstances](#) di AWS SDK untuk referensi API Kotlin.

## StartInstances

Contoh kode berikut menunjukkan cara menggunakan `StartInstances`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).



```
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.waitForInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}
```

- Untuk detail API, lihat [StartInstances](#) di AWS SDK untuk referensi API Kotlin.

## StopInstances

Contoh kode berikut menunjukkan cara menggunakan `StopInstances`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.waitForInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}

```

- Untuk detail API, lihat [StopInstances](#) di AWS SDK untuk referensi API Kotlin.

## TerminateInstances

Contoh kode berikut menunjukkan cara menggunakan `TerminateInstances`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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}")
        }
    }
}
}

```

- Untuk detail API, lihat [TerminateInstances](#) di AWS SDK untuk referensi API Kotlin.

## Contoh Amazon ECR menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Amazon ECR.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

### Memulai

#### Halo Amazon ECR

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon ECR.

#### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [listImages](#) di AWS SDK untuk referensi API Kotlin.

## Topik

- [Hal-hal mendasar](#)
- [Tindakan](#)

## Hal-hal mendasar

### Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara:

- Buat repositori Amazon ECR.
- Tetapkan kebijakan repositori.
- Ambil URIs repositori.
- Dapatkan token otorisasi Amazon ECR.

- Tetapkan kebijakan siklus hidup untuk repositori Amazon ECR.
- Dorong gambar Docker ke repositori Amazon ECR.
- Verifikasi keberadaan gambar di repositori Amazon ECR.
- Buat daftar repositori Amazon ECR untuk akun Anda dan dapatkan detailnya.
- Hapus repositori Amazon ECR.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

Jalankan skenario interaktif yang menunjukkan fitur Amazon ECR.

```
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.")
        }
    }
}

```

```

    }
}
}

```

Kelas pembungkus untuk metode Amazon ECR SDK.

```

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)
        }
    }
}

```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [CreateRepository](#)

- [DeleteRepository](#)
- [DescribeImages](#)
- [DescribeRepositories](#)
- [GetAuthorizationToken](#)
- [GetRepositoryPolicy](#)
- [SetRepositoryPolicy](#)
- [StartLifecyclePolicyPreview](#)

## Tindakan

### CreateRepository

Contoh kode berikut menunjukkan cara menggunakan `CreateRepository`.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
/**
 * 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
}
}
```

- Untuk detail API, lihat [CreateRepository](#) di AWS SDK untuk referensi API Kotlin.

## DeleteRepository

Contoh kode berikut menunjukkan cara menggunakan `DeleteRepository`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
/**
 * 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")
    }
}

```

- Untuk detail API, lihat [DeleteRepository](#) di AWS SDK untuk referensi API Kotlin.

## DescribeImages

Contoh kode berikut menunjukkan cara menggunakan `DescribeImages`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

/**
 * 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.")
            }
        }
    }
}

```

- Untuk detail API, lihat [DescribeImages](#) di AWS SDK untuk referensi API Kotlin.

## DescribeRepositories

Contoh kode berikut menunjukkan cara menggunakan `DescribeRepositories`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

/**
 * 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 ""
        }
    }
}

```

- Untuk detail API, lihat [DescribeRepositories](#) di AWS SDK untuk referensi API Kotlin.

## GetAuthorizationToken

Contoh kode berikut menunjukkan cara menggunakan `GetAuthorizationToken`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

/**
 * 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.")
        }
    }
}

```

- Untuk detail API, lihat [GetAuthorizationToken](#) di AWS SDK untuk referensi API Kotlin.

## GetRepositoryPolicy

Contoh kode berikut menunjukkan cara menggunakan `GetRepositoryPolicy`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

/**
 * 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
}
}
```

- Untuk detail API, lihat [GetRepositoryPolicy](#) di AWS SDK untuk referensi API Kotlin.

## PushImageCmd

Contoh kode berikut menunjukkan cara menggunakan `PushImageCmd`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
/**
 * 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)
    }
}
}

```

- Untuk detail API, lihat [PushImageCmd](#) di AWS SDK untuk referensi API Kotlin.

## SetRepositoryPolicy

Contoh kode berikut menunjukkan cara menggunakan `SetRepositoryPolicy`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
/**
 * 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.")
        }
    }
}

```

- Untuk detail API, lihat [SetRepositoryPolicy](#) di AWS SDK untuk referensi API Kotlin.

## StartLifecyclePolicyPreview

Contoh kode berikut menunjukkan cara menggunakan `StartLifecyclePolicyPreview`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

/**
 * 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.")
    }
}
}
```

- Untuk detail API, lihat [StartLifecyclePolicyPreview](#) di AWS SDK untuk referensi API Kotlin.

## OpenSearch Contoh layanan menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan OpenSearch Service.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Tindakan](#)

### Tindakan

#### **CreateDomain**

Contoh kode berikut menunjukkan cara menggunakan `CreateDomain`.



## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
}
}
```

- Untuk detail API, lihat [CreateDomain](#) di AWS SDK untuk referensi API Kotlin.

## DeleteDomain

Contoh kode berikut menunjukkan cara menggunakan `DeleteDomain`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).


```
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.")
    }
}
```

- Untuk detail API, lihat [DeleteDomain](#) di AWS SDK untuk referensi API Kotlin.

## ListDomainNames

Contoh kode berikut menunjukkan cara menggunakan `ListDomainNames`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).


```
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)
        }
    }
}
```

- Untuk detail API, lihat [ListDomainNames](#) di AWS SDK untuk referensi API Kotlin.

## UpdateDomainConfig

Contoh kode berikut menunjukkan cara menggunakan `UpdateDomainConfig`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun updateSpecificDomain(domainNameVal: String?) {
    val clusterConfig0b =
    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())
}
}
```

- Untuk detail API, lihat [UpdateDomainConfig](#) di AWS SDK untuk referensi API Kotlin.

## EventBridge contoh menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin with. EventBridge

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Memulai

Halo EventBridge

Contoh kode berikut menunjukkan cara untuk mulai menggunakan EventBridge.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListEventBuses](#) di AWS SDK untuk referensi API Kotlin.

## Topik

- [Hal-hal mendasar](#)
- [Tindakan](#)

## Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara:

- Buat aturan dan tambahkan target ke dalamnya.
- Aktifkan dan nonaktifkan aturan.
- Daftar dan perbarui aturan dan target.
- Kirim acara, lalu bersihkan sumber daya.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
/*
```

```
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 = targets0b
        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
}
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [DeleteRule](#)
  - [DescribeRule](#)
  - [DisableRule](#)
  - [EnableRule](#)
  - [ListRuleNamesByTarget](#)
  - [ListRules](#)
  - [ListTargetsByRule](#)
  - [PutEvents](#)
  - [PutRule](#)
  - [PutTargets](#)

## Tindakan

### DeleteRule

Contoh kode berikut menunjukkan cara menggunakan `DeleteRule`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [DeleteRule](#) di AWS SDK untuk referensi API Kotlin.

## DescribeRule

Contoh kode berikut menunjukkan cara menggunakan `DescribeRule`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
}
}
```

- Untuk detail API, lihat [DescribeRule](#) di AWS SDK untuk referensi API Kotlin.

## DisableRule

Contoh kode berikut menunjukkan cara menggunakan `DisableRule`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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)
        }
    }
}

```

- Untuk detail API, lihat [DisableRule](#) di AWS SDK untuk referensi API Kotlin.

## EnableRule

Contoh kode berikut menunjukkan cara menggunakan `EnableRule`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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)
    }
}
}

```

- Untuk detail API, lihat [EnableRule](#) di AWS SDK untuk referensi API Kotlin.

## ListRuleNamesByTarget

Contoh kode berikut menunjukkan cara menggunakan `ListRuleNamesByTarget`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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")
        }
    }
}
}


```

- Untuk detail API, lihat [ListRuleNamesByTarget](#) di AWS SDK untuk referensi API Kotlin.

## ListRules

Contoh kode berikut menunjukkan cara menggunakan `ListRules`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListRules](#) di AWS SDK untuk referensi API Kotlin.

## ListTargetsByRule

Contoh kode berikut menunjukkan cara menggunakan `ListTargetsByRule`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
}

```

- Untuk detail API, lihat [ListTargetsByRule](#) di AWS SDK untuk referensi API Kotlin.

## PutEvents

Contoh kode berikut menunjukkan cara menggunakan PutEvents.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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)
    }
}

```

- Untuk detail API, lihat [PutEvents](#) di AWS SDK untuk referensi API Kotlin.

## PutRule

Contoh kode berikut menunjukkan cara menggunakan PutRule.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat aturan terjadwal.

```

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}")
}
}

```

Buat aturan yang dipicu saat objek ditambahkan ke bucket Amazon Simple Storage Service.

```

// 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}")
    }
}

```

- Untuk detail API, lihat [PutRule](#) di AWS SDK untuk referensi API Kotlin.

## PutTargets

Contoh kode berikut menunjukkan cara menggunakan PutTargets.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
// 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 targets0b = mutableListOf<Target>()
    targets0b.add(myTarget)

    val request =
        PutTargetsRequest {
            eventBusName = null
            targets = targets0b
            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.")
    }
}
```



Tambahkan transformator input ke target untuk aturan.

```
suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerObj =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\""
        }

    val target =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformerObj
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(target)
            eventBusName = null
        }


    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}
```

- Untuk detail API, lihat [PutTargets](#) di AWS SDK untuk referensi API Kotlin.

## RemoveTargets

Contoh kode berikut menunjukkan cara menggunakan `RemoveTargets`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [RemoveTargets](#) di AWS SDK untuk referensi API Kotlin.

## AWS Glue contoh menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin with. AWS Glue

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Hal-hal mendasar](#)
- [Tindakan](#)

## Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara:

- Buat crawler yang merayapi bucket Amazon S3 publik dan membuat database metadata berformat CSV.
- Daftar informasi tentang database dan tabel di situs Anda AWS Glue Data Catalog.
- Buat pekerjaan untuk mengekstrak data CSV dari bucket S3, mengubah data, dan memuat output berformat JSON ke bucket S3 lain.
- Buat daftar informasi tentang menjalankan pekerjaan, melihat data yang diubah, dan membersihkan sumber daya.

Untuk informasi selengkapnya, lihat [Tutorial: Memulai AWS Glue Studio](#).

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")  
    }  
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [CreateCrawler](#)
  - [CreateJob](#)
  - [DeleteCrawler](#)
  - [DeleteDatabase](#)
  - [DeleteJob](#)
  - [DeleteTable](#)
  - [GetCrawler](#)
  - [GetDatabase](#)
  - [GetDatabases](#)
  - [GetJob](#)
  - [GetJobRun](#)
  - [GetJobRuns](#)
  - [GetTables](#)
  - [ListJobs](#)
  - [StartCrawler](#)
  - [StartJobRun](#)

## Tindakan

### **CreateCrawler**

Contoh kode berikut menunjukkan cara menggunakan `CreateCrawler`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

```
}  
}
```

- Untuk detail API, lihat [CreateCrawler](#) di AWS SDK untuk referensi API Kotlin.

## GetCrawler

Contoh kode berikut menunjukkan cara menggunakan `GetCrawler`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).


```
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")  
    }  
}
```

- Untuk detail API, lihat [GetCrawler](#) di AWS SDK untuk referensi API Kotlin.

## GetDatabase

Contoh kode berikut menunjukkan cara menggunakan `GetDatabase`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [GetDatabase](#) di AWS SDK untuk referensi API Kotlin.

**StartCrawler**

Contoh kode berikut menunjukkan cara menggunakan `StartCrawler`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.")
    }
}
```

- Untuk detail API, lihat [StartCrawler](#) di AWS SDK untuk referensi API Kotlin.

## Contoh IAM menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan IAM.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Hal-hal mendasar](#)
- [Tindakan](#)

## Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut menunjukkan cara membuat pengguna dan mengambil peran.

**⚠ Warning**

Untuk menghindari risiko keamanan, jangan gunakan pengguna IAM untuk otentikasi saat mengembangkan perangkat lunak yang dibuat khusus atau bekerja dengan data nyata. Sebaliknya, gunakan federasi dengan penyedia identitas seperti [AWS IAM Identity Center](#).

- Buat pengguna tanpa izin.
- Buat peran yang memberikan izin untuk mencantumkan bucket Amazon S3 untuk akun tersebut.
- Tambahkan kebijakan agar pengguna dapat mengambil peran tersebut.
- Asumsikan peran dan daftar bucket S3 menggunakan kredensial sementara, lalu bersihkan sumber daya.

## SDK untuk Kotlin

**i Note**

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat fungsi yang membungkus tindakan pengguna IAM.

```
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)
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## Tindakan

### AttachRolePolicy

Contoh kode berikut menunjukkan cara menggunakan `AttachRolePolicy`.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
}

```

- Untuk detail API, lihat [AttachRolePolicy](#) di AWS SDK untuk referensi API Kotlin.

## CreateAccessKey

Contoh kode berikut menunjukkan cara menggunakan `CreateAccessKey`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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()
    }
}
```

- Untuk detail API, lihat [CreateAccessKey](#) di AWS SDK untuk referensi API Kotlin.

## CreateAccountAlias

Contoh kode berikut menunjukkan cara menggunakan `CreateAccountAlias`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [CreateAccountAlias](#) di AWS SDK untuk referensi API Kotlin.

## CreatePolicy

Contoh kode berikut menunjukkan cara menggunakan `CreatePolicy`.



## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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()
    }
}
```

- Untuk detail API, lihat [CreatePolicy](#) di AWS SDK untuk referensi API Kotlin.

## CreateUser

Contoh kode berikut menunjukkan cara menggunakan `CreateUser`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}
```

- Untuk detail API, lihat [CreateUser](#) di AWS SDK untuk referensi API Kotlin.

## DeleteAccessKey

Contoh kode berikut menunjukkan cara menggunakan `DeleteAccessKey`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [DeleteAccessKey](#) di AWS SDK untuk referensi API Kotlin.

## DeleteAccountAlias

Contoh kode berikut menunjukkan cara menggunakan `DeleteAccountAlias`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [DeleteAccountAlias](#) di AWS SDK untuk referensi API Kotlin.

## DeletePolicy

Contoh kode berikut menunjukkan cara menggunakan `DeletePolicy`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {
    val request =
        DeletePolicyRequest {
            policyArn = policyARNVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- Untuk detail API, lihat [DeletePolicy](#) di AWS SDK untuk referensi API Kotlin.

## DeleteUser

Contoh kode berikut menunjukkan cara menggunakan `DeleteUser`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [DeleteUser](#) di AWS SDK untuk referensi API Kotlin.

## DetachRolePolicy

Contoh kode berikut menunjukkan cara menggunakan `DetachRolePolicy`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
}
}

```

- Untuk detail API, lihat [DetachRolePolicy](#) di AWS SDK untuk referensi API Kotlin.

## GetPolicy

Contoh kode berikut menunjukkan cara menggunakan `GetPolicy`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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}")
    }
}

```

- Untuk detail API, lihat [GetPolicy](#) di AWS SDK untuk referensi API Kotlin.

## ListAccessKeys

Contoh kode berikut menunjukkan cara menggunakan `ListAccessKeys`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListAccessKeys](#) di AWS SDK untuk referensi API Kotlin.

## ListAccountAliases

Contoh kode berikut menunjukkan cara menggunakan `ListAccountAliases`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun listAliases() {
    iamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- Untuk detail API, lihat [ListAccountAliases](#) di AWS SDK untuk referensi API Kotlin.

## ListUsers

Contoh kode berikut menunjukkan cara menggunakan `ListUsers`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListUsers](#) di AWS SDK untuk referensi API Kotlin.



## UpdateUser

Contoh kode berikut menunjukkan cara menggunakan `UpdateUser`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [UpdateUser](#) di AWS SDK untuk referensi API Kotlin.

## AWS IoT contoh menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin with. AWS IoT

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.


Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Memulai

Halo AWS IoT

Contoh kode berikut menunjukkan cara untuk mulai menggunakan AWS IoT.

SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

```
    }  
  }  
}
```

- Untuk detail API, lihat [ListThings](#) di AWS SDK untuk referensi API Kotlin.

## Topik

- [Hal-hal mendasar](#)
- [Tindakan](#)

## Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut menunjukkan cara bekerja dengan manajemen AWS IoT perangkat.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.ByteString
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")
    }
}
}


```

## Tindakan

### **AttachThingPrincipal**

Contoh kode berikut menunjukkan cara menggunakan `AttachThingPrincipal`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.")
    }
}
```

- Untuk detail API, lihat [AttachThingPrincipal](#) di AWS SDK untuk referensi API Kotlin.

## CreateKeysAndCertificate

Contoh kode berikut menunjukkan cara menggunakan `CreateKeysAndCertificate`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
}
}
```

- Untuk detail API, lihat [CreateKeysAndCertificate](#) di AWS SDK untuk referensi API Kotlin.

## CreateThing

Contoh kode berikut menunjukkan cara menggunakan `CreateThing`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal")
    }
}
```

- Untuk detail API, lihat [CreateThing](#) di AWS SDK untuk referensi API Kotlin.

## CreateTopicRule

Contoh kode berikut menunjukkan cara menggunakan `CreateTopicRule`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.")
}
}
```

- Untuk detail API, lihat [CreateTopicRule](#) di AWS SDK untuk referensi API Kotlin.

## DeleteCertificate

Contoh kode berikut menunjukkan cara menggunakan `DeleteCertificate`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.")
    }
}
```

- Untuk detail API, lihat [DeleteCertificate](#) di AWS SDK untuk referensi API Kotlin.

## DeleteThing

Contoh kode berikut menunjukkan cara menggunakan `DeleteThing`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- Untuk detail API, lihat [DeleteThing](#) di AWS SDK untuk referensi API Kotlin.

## DescribeEndpoint

Contoh kode berikut menunjukkan cara menggunakan `DescribeEndpoint`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}

```

- Untuk detail API, lihat [DescribeEndpoint](#) di AWS SDK untuk referensi API Kotlin.

## DescribeThing

Contoh kode berikut menunjukkan cara menggunakan `DescribeThing`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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}")
    }
}

```

- Untuk detail API, lihat [DescribeThing](#) di AWS SDK untuk referensi API Kotlin.

## DetachThingPrincipal

Contoh kode berikut menunjukkan cara menggunakan `DetachThingPrincipal`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [DetachThingPrincipal](#) di AWS SDK untuk referensi API Kotlin.

## ListCertificates

Contoh kode berikut menunjukkan cara menggunakan `ListCertificates`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).


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

- Untuk detail API, lihat [ListCertificates](#) di AWS SDK untuk referensi API Kotlin.

## SearchIndex

Contoh kode berikut menunjukkan cara menggunakan `SearchIndex`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [SearchIndex](#) di AWS SDK untuk referensi API Kotlin.

## UpdateThing

Contoh kode berikut menunjukkan cara menggunakan `UpdateThing`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.")
}
}
```

- Untuk detail API, lihat [UpdateThing](#) di AWS SDK untuk referensi API Kotlin.

## AWS IoT data contoh menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin with. AWS IoT data

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Tindakan](#)

## Tindakan

### **GetThingShadow**

Contoh kode berikut menunjukkan cara menggunakan `GetThingShadow`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [GetThingShadow](#) di AWS SDK untuk referensi API Kotlin.

## UpdateThingShadow

Contoh kode berikut menunjukkan cara menggunakan `UpdateThingShadow`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.")
}
}
```

- Untuk detail API, lihat [UpdateThingShadow](#) di AWS SDK untuk referensi API Kotlin.

## Contoh Amazon Keyspaces menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Amazon Keyspaces.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.


Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

### Memulai

#### Halo Amazon Keyspaces

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon Keyspaces.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListKeyspaces](#) di AWS SDK untuk referensi API Kotlin.

## Topik

- [Hal-hal mendasar](#)

- [Tindakan](#)

## Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara:

- Buat keyspace dan tabel. Skema tabel menyimpan data film dan mengaktifkan point-in-time pemulihan.
- Connect ke keyspace menggunakan koneksi TLS aman dengan otentikasi SiGv4.
- Kueri tabel. Tambahkan, ambil, dan perbarui data film.
- Perbarui tabel. Tambahkan kolom untuk melacak film yang ditonton.
- Kembalikan tabel ke keadaan sebelumnya dan bersihkan sumber daya.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
/**
```

```
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 schemaDefinition0b =
    SchemaDefinition {
        partitionKeys = keyList
        allColumns = collList
    }

val timeRecovery =
    PointInTimeRecovery {
        status = PointInTimeRecoveryStatus.Enabled
    }

val tableRequest =
    CreateTableRequest {
        keyspaceName = keySpaceVal
        tableName = tableNameVal
        schemaDefinition = schemaDefinition0b
        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}")
    }
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [CreateKeyspace](#)
  - [CreateTable](#)
  - [DeleteKeyspace](#)
  - [DeleteTable](#)
  - [GetKeyspace](#)
  - [GetTable](#)
  - [ListKeyspaces](#)

- [ListTables](#)
- [RestoreTable](#)
- [UpdateTable](#)

## Tindakan

### CreateKeyspace

Contoh kode berikut menunjukkan cara menggunakan `CreateKeyspace`.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [CreateKeyspace](#) di AWS SDK untuk referensi API Kotlin.

### CreateTable

Contoh kode berikut menunjukkan cara menggunakan `CreateTable`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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 schemaDefinition0b =
    SchemaDefinition {
        partitionKeys = keyList
        allColumns = colList
    }

val timeRecovery =
    PointInTimeRecovery {
        status = PointInTimeRecoveryStatus.Enabled
    }

val tableRequest =
    CreateTableRequest {
        keyspaceName = keySpaceVal
        tableName = tableNameVal
        schemaDefinition = schemaDefinition0b
        pointInTimeRecovery = timeRecovery
    }

KeyspacesClient { region = "us-east-1" }.use { keyClient ->
    val response = keyClient.createTable(tableRequest)
    println("The table ARN is ${response.resourceArn}")
}
}
```

- Untuk detail API, lihat [CreateTable](#) di AWS SDK untuk referensi API Kotlin.

## DeleteKeyspace

Contoh kode berikut menunjukkan cara menggunakan `DeleteKeyspace`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}
```

- Untuk detail API, lihat [DeleteKeyspace](#) di AWS SDK untuk referensi API Kotlin.

## DeleteTable

Contoh kode berikut menunjukkan cara menggunakan `DeleteTable`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun deleteTable(
```

```
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}
```

- Untuk detail API, lihat [DeleteTable](#) di AWS SDK untuk referensi API Kotlin.

## GetKeyspace

Contoh kode berikut menunjukkan cara menggunakan `GetKeyspace`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [GetKeyspace](#) di AWS SDK untuk referensi API Kotlin.

## GetTable

Contoh kode berikut menunjukkan cara menggunakan `GetTable`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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



```

    }
  }
}

```

- Untuk detail API, lihat [GetTable](#) di AWS SDK untuk referensi API Kotlin.

## ListKeyspaces

Contoh kode berikut menunjukkan cara menggunakan `ListKeyspaces`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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}")
            }
    }
}


```

- Untuk detail API, lihat [ListKeyspaces](#) di AWS SDK untuk referensi API Kotlin.

## ListTables

Contoh kode berikut menunjukkan cara menggunakan `ListTables`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListTables](#) di AWS SDK untuk referensi API Kotlin.

## RestoreTable

Contoh kode berikut menunjukkan cara menggunakan `RestoreTable`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}

```

- Untuk detail API, lihat [RestoreTable](#) di AWS SDK untuk referensi API Kotlin.

## UpdateTable

Contoh kode berikut menunjukkan cara menggunakan `UpdateTable`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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)
    }
}
```

- Untuk detail API, lihat [UpdateTable](#) di AWS SDK untuk referensi API Kotlin.

## AWS KMS contoh menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin with. AWS KMS

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik


- [Tindakan](#)

### Tindakan

#### **CreateAlias**

Contoh kode berikut menunjukkan cara menggunakan `CreateAlias`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [CreateAlias](#) di AWS SDK untuk referensi API Kotlin.

## CreateGrant

Contoh kode berikut menunjukkan cara menggunakan `CreateGrant`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun createNewGrant(
```

```

    keyIdVal: String?,
    granteePrincipalVal: String?,
    operation: String,
): String? {
    val operationObj = GrantOperation.fromValue(operation)
    val grantOperationList = ArrayList<GrantOperation>()
    grantOperationList.add(operationObj)

    val request =
        CreateGrantRequest {
            keyId = keyIdVal
            granteePrincipal = granteePrincipalVal
            operations = grantOperationList
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.createGrant(request)
        return response.grantId
    }
}

```

- Untuk detail API, lihat [CreateGrant](#) di AWS SDK untuk referensi API Kotlin.

## CreateKey

Contoh kode berikut menunjukkan cara menggunakan `CreateKey`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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
    }
}

```

- Untuk detail API, lihat [CreateKey](#) di AWS SDK untuk referensi API Kotlin.

## Decrypt

Contoh kode berikut menunjukkan cara menggunakan `Decrypt`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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)
    }
}
}

```

- Untuk detail API, lihat [Mendekripsi](#) di AWS SDK untuk referensi API Kotlin.

## DescribeKey

Contoh kode berikut menunjukkan cara menggunakan `DescribeKey`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

suspend fun describeSpecificKey(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}")
}
}
```

- Untuk detail API, lihat [DescribeKey](#) di AWS SDK untuk referensi API Kotlin.

## DisableKey

Contoh kode berikut menunjukkan cara menggunakan `DisableKey`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [DisableKey](#) di AWS SDK untuk referensi API Kotlin.

## EnableKey

Contoh kode berikut menunjukkan cara menggunakan `EnableKey`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.")
    }
}
```

- Untuk detail API, lihat [EnableKey](#) di AWS SDK untuk referensi API Kotlin.

## Encrypt

Contoh kode berikut menunjukkan cara menggunakan `Encrypt`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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)
    }
}
```

- Untuk detail API, lihat [Enkripsi](#) di AWS SDK untuk referensi API Kotlin.

## ListAliases

Contoh kode berikut menunjukkan cara menggunakan `ListAliases`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListAliases](#) di AWS SDK untuk referensi API Kotlin.

## ListGrants

Contoh kode berikut menunjukkan cara menggunakan `ListGrants`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
        }
    }
}

```

- Untuk detail API, lihat [ListGrants](#) di AWS SDK untuk referensi API Kotlin.

## ListKeys

Contoh kode berikut menunjukkan cara menggunakan `ListKeys`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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}")
        }
    }
}

```

- Untuk detail API, lihat [ListKeys](#) di AWS SDK untuk referensi API Kotlin.

## Contoh Lambda menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Lambda.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Hal-hal mendasar](#)
- [Tindakan](#)
- [Skenario](#)

## Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara:

- Buat peran IAM dan fungsi Lambda, lalu unggah kode handler.
- Panggil fungsi dengan satu parameter dan dapatkan hasil.
- Perbarui kode fungsi dan konfigurasi dengan variabel lingkungan.

- Panggil fungsi dengan parameter baru dan dapatkan hasil. Tampilkan log eksekusi yang dikembalikan.
- Buat daftar fungsi untuk akun Anda, lalu bersihkan sumber daya.

Untuk informasi selengkapnya, lihat [Membuat fungsi Lambda dengan konsol](#).

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.waitForFunctionActive {
            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")
    }
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Memohon](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

# Tindakan

## CreateFunction

Contoh kode berikut menunjukkan cara menggunakan `CreateFunction`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}
```

- Untuk detail API, lihat [CreateFunction](#) di AWS SDK untuk referensi API Kotlin.

## DeleteFunction

Contoh kode berikut menunjukkan cara menggunakan `DeleteFunction`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun dellambdaFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- Untuk detail API, lihat [DeleteFunction](#) di AWS SDK untuk referensi API Kotlin.

## Invoke

Contoh kode berikut menunjukkan cara menggunakan `Invoke`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [Memanggil](#) di AWS SDK untuk referensi API Kotlin.

## Skenario

Membuat aplikasi nirserver untuk mengelola foto

Contoh kode berikut menunjukkan cara membuat aplikasi tanpa server yang memungkinkan pengguna mengelola foto menggunakan label.

SDK untuk Kotlin

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

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

## MediaConvert contoh menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin with. MediaConvert

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Tindakan](#)

## Tindakan

### **CreateJob**

Contoh kode berikut menunjukkan cara menggunakan `CreateJob`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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> = mutableListOfOf()
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

```

```

H264ParControl.InitializeFromSource
H264QualityTuningLevel.SinglePass
{ qvbrQualityLevel = 8 }

H264FramerateControl.InitializeFromSource
    parControl =
    qualityTuningLevel =
    qvbrSettings = H264QvbrSettings
    codecLevel = H264CodecLevel.Auto
    codecProfile = H264CodecProfile.Main
    maxBitrate = 2400000
    framerateControl =
    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 =
    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
            }
        }
    }

```

```

        H264DynamicSubGop.Static
        H264FieldEncoding.Paff
        H264SceneChangeDetect.Enabled
        H264FramerateConversionAlgorithm.DuplicateDrop
        H264EntropyEncoding.Cabac
        H264UnregisteredSeiTimecode.Disabled
        H264AdaptiveQuantization.High
        H264SpatialAdaptiveQuantization.Enabled
        H264TemporalAdaptiveQuantization.Enabled
        H264FlickerAdaptiveQuantization.Disabled
        H264InterlaceMode.Progressive
    }
    }
    audioDescriptions = listOf(audio1)
}
} catch (ex: MediaConvertException) {
    println(ex.toString())
    exitProcess(0)
}
return output
}
dynamicSubGop =
fieldEncoding =
sceneChangeDetect =
minIInterval = 0
telecine = H264Telecine.None
framerateConversionAlgorithm =
entropyEncoding =
slices = 1
unregisteredSeiTimecode =
repeatPps = H264RepeatPps.Disabled
adaptiveQuantization =
spatialAdaptiveQuantization =
temporalAdaptiveQuantization =
flickerAdaptiveQuantization =
softness = 0
interlaceMode =

```

- Untuk detail API, lihat [CreateJob](#) di AWS SDK untuk referensi API Kotlin.

## GetJob

Contoh kode berikut menunjukkan cara menggunakan `GetJob`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}.")
}
```

- Untuk detail API, lihat [GetJob](#) di AWS SDK untuk referensi API Kotlin.

## ListJobs

Contoh kode berikut menunjukkan cara menggunakan `ListJobs`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListJobs](#) di AWS SDK untuk referensi API Kotlin.

## Amazon Pinpoint contoh menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Amazon Pinpoint.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Tindakan](#)

### Tindakan

#### CreateApp

Contoh kode berikut menunjukkan cara menggunakan `CreateApp`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun createApplication(applicationName: String?): String? {
    val createApplicationRequest0b =
        CreateApplicationRequest {
            name = applicationName
        }


    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.createApp(
                CreateAppRequest {
                    createApplicationRequest = createApplicationRequest0b
                },
            )
        return result.applicationResponse?.id
    }
}
```

- Untuk detail API, lihat [CreateApp](#) di AWS SDK untuk referensi API Kotlin.

## CreateCampaign

Contoh kode berikut menunjukkan cara menggunakan `CreateCampaign`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun createPinCampaign(
    appId: String,
    segmentIdVal: String,
) {
    val schedule0b =
        Schedule {
            startTime = "IMMEDIATE"
        }

    val defaultMessage0b =
        Message {
            action = Action.OpenApp
            body = "My message body"
            title = "My message title"
        }

    val messageConfiguration0b =
        MessageConfiguration {
            defaultMessage = defaultMessage0b
        }

    val writeCampaign =
        WriteCampaignRequest {
            description = "My description"
            schedule = schedule0b
            name = "MyCampaign"
            segmentId = segmentIdVal
            messageConfiguration = messageConfiguration0b
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result: CreateCampaignResponse =
            pinpoint.createCampaign(
                CreateCampaignRequest {
                    applicationId = appId
                    writeCampaignRequest = writeCampaign
                },
            )
        println("Campaign ID is ${result.campaignResponse?.id}")
    }
}
```

- Untuk detail API, lihat [CreateCampaign](#) di AWS SDK untuk referensi API Kotlin.

## CreateSegment

Contoh kode berikut menunjukkan cara menggunakan `CreateSegment`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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 writeSegmentRequest0b =
        WriteSegmentRequest {
            name = "MySegment101"
            dimensions = dimensions0b
        }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val createSegmentResult: CreateSegmentResponse =
            pinpoint.createSegment(
                CreateSegmentRequest {
                    applicationId = applicationIdVal
                    writeSegmentRequest = writeSegmentRequest0b
                },
            )
        println("Segment ID is ${createSegmentResult.segmentResponse?.id}")
        return createSegmentResult.segmentResponse?.id
    }
}

```

- Untuk detail API, lihat [CreateSegment](#) di AWS SDK untuk referensi API Kotlin.

## DeleteApp

Contoh kode berikut menunjukkan cara menggunakan `DeleteApp`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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.")
    }
}

```

- Untuk detail API, lihat [DeleteApp](#) di AWS SDK untuk referensi API Kotlin.

## DeleteEndpoint

Contoh kode berikut menunjukkan cara menggunakan `DeleteEndpoint`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

suspend fun deletePinEndpoint(
    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")
    }
}

```

- Untuk detail API, lihat [DeleteEndpoint](#) di AWS SDK untuk referensi API Kotlin.

## GetEndpoint

Contoh kode berikut menunjukkan cara menggunakan `GetEndpoint`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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)
    }
}
```

- Untuk detail API, lihat [GetEndpoint](#) di AWS SDK untuk referensi API Kotlin.

## GetSegments

Contoh kode berikut menunjukkan cara menggunakan `GetSegments`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [GetSegments](#) di AWS SDK untuk referensi API Kotlin.

## SendMessages

Contoh kode berikut menunjukkan cara menggunakan `SendMessages`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [SendMessages](#) di AWS SDK untuk referensi API Kotlin.

## Contoh Amazon RDS menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Amazon RDS.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

### Topik

- [Hal-hal mendasar](#)
- [Tindakan](#)
- [Skenario](#)

## Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara:

- Membuat grup parameter basis data kustom dan mengatur nilai parameter.
- Membuat instans basis data yang dikonfigurasi untuk menggunakan grup parameter. Instans basis data juga berisi basis data.
- Mengambil cuplikan instans.
- Menghapus instans dan grup parameter.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
/**
```

```
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
${engineObj.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
    }
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [Buat DBInstance](#)
  - [Buat DBParameter Grup](#)
  - [Buat DBSnapshot](#)
  - [Hapus DBInstance](#)
  - [Hapus DBParameter Grup](#)
  - [Jelaskan DBEngine Versi](#)
  - [Jelaskan DBInstances](#)
  - [Jelaskan DBParameter Grup](#)
  - [Jelaskan DBParameters](#)
  - [Jelaskan DBSnapshots](#)
  - [DescribeOrderableDBInstancePilihan](#)
  - [Ubah DBParameter Grup](#)

# Tindakan

## CreateDBInstance

Contoh kode berikut menunjukkan cara menggunakan `CreateDBInstance`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun createDatabaseInstance(
    dbInstanceIdentifierVal: String?,
    dbNameVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNameVal
            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!")
}
```

- Untuk detail API, lihat [Membuat DBInstance](#) di AWS SDK untuk referensi API Kotlin.

## DeleteDBInstance

Contoh kode berikut menunjukkan cara menggunakan `DeleteDBInstance`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [Menghapus DBInstance](#) di AWS SDK untuk referensi API Kotlin.

## DescribeAccountAttributes

Contoh kode berikut menunjukkan cara menggunakan `DescribeAccountAttributes`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
        }
    }
}

```

- Untuk detail API, lihat [DescribeAccountAttributes](#) di AWS SDK untuk referensi API Kotlin.

## DescribeDBInstances

Contoh kode berikut menunjukkan cara menggunakan `DescribeDBInstances`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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}")
        }
    }
}

```

- Untuk detail API, lihat [Menjelaskan DBInstances](#) di AWS SDK untuk referensi API Kotlin.

## ModifyDBInstance

Contoh kode berikut menunjukkan cara menggunakan `ModifyDBInstance`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [Memodifikasi DBInstance](#) di AWS SDK untuk referensi API Kotlin.

## Skenario

Buat pelacak butir kerja Aurora Nirserver

Contoh kode berikut menunjukkan cara membuat aplikasi web yang melacak item pekerjaan dalam database Amazon Aurora Tanpa Server dan menggunakan Amazon Simple Email Service (Amazon SES) untuk mengirim laporan.



## SDK untuk Kotlin

Menunjukkan cara membuat aplikasi web yang melacak dan melaporkan butir kerja yang tersimpan dalam basis data Amazon RDS.

Untuk kode sumber lengkap dan petunjuk tentang cara menyiapkan Spring REST API yang menanyakan data Amazon Aurora Tanpa Server dan untuk digunakan oleh aplikasi React, lihat contoh lengkapnya di [GitHub](#)

Layanan yang digunakan dalam contoh ini

- Aurora
- Amazon RDS
- Layanan Data Amazon RDS
- Amazon SES

## Contoh Amazon RDS Data Service menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Amazon RDS Data Service.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Skenario](#)

## Skenario

Buat pelacak butir kerja Aurora Nirserver

Contoh kode berikut menunjukkan cara membuat aplikasi web yang melacak item pekerjaan dalam database Amazon Aurora Tanpa Server dan menggunakan Amazon Simple Email Service (Amazon SES) untuk mengirim laporan.

## SDK untuk Kotlin

Menunjukkan cara membuat aplikasi web yang melacak dan melaporkan butir kerja yang tersimpan dalam basis data Amazon RDS.

Untuk kode sumber lengkap dan petunjuk tentang cara menyiapkan Spring REST API yang menanyakan data Amazon Aurora Tanpa Server dan untuk digunakan oleh aplikasi React, lihat contoh lengkapnya di [GitHub](#)

Layanan yang digunakan dalam contoh ini

- Aurora
- Amazon RDS
- Layanan Data Amazon RDS
- Amazon SES

## Contoh Amazon Redshift menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Amazon Redshift.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

### Topik

- [Tindakan](#)
- [Skenario](#)

## Tindakan

### CreateCluster

Contoh kode berikut menunjukkan cara menggunakan `CreateCluster`.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat cluster.

```
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}")
    }
}
```

- Untuk detail API, lihat [CreateCluster](#) di AWS SDK untuk referensi API Kotlin.

## DeleteCluster

Contoh kode berikut menunjukkan cara menggunakan `DeleteCluster`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus klaster .

```
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}")
    }
}
```

- Untuk detail API, lihat [DeleteCluster](#) di AWS SDK untuk referensi API Kotlin.

## DescribeClusters

Contoh kode berikut menunjukkan cara menggunakan `DescribeClusters`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

## Jelaskan cluster.

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

- Untuk detail API, lihat [DescribeClusters](#) di AWS SDK untuk referensi API Kotlin.

## ModifyCluster

Contoh kode berikut menunjukkan cara menggunakan `ModifyCluster`.

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

## Memodifikasi cluster.

```
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",
        )
    }
}
```

- Untuk detail API, lihat [ModifyCluster](#) di AWS SDK untuk referensi API Kotlin.

## Skenario

Buat aplikasi web untuk melacak data Amazon Redshift

Contoh kode berikut menunjukkan cara membuat aplikasi web yang melacak dan melaporkan item pekerjaan menggunakan database Amazon Redshift.

SDK untuk Kotlin

Menunjukkan cara membuat aplikasi web yang melacak dan melaporkan item pekerjaan yang disimpan dalam database Amazon Redshift.

Untuk kode sumber lengkap dan petunjuk tentang cara menyiapkan Spring REST API yang menanyakan data Amazon Redshift dan untuk digunakan oleh aplikasi React, lihat contoh lengkapnya di [GitHub](#)

Layanan yang digunakan dalam contoh ini

- Amazon Redshift
- Amazon SES

## Contoh Rekognition Amazon menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Amazon Rekognition.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Tindakan](#)
- [Skenario](#)

## Tindakan

### CompareFaces

Contoh kode berikut menunjukkan cara menggunakan `CompareFaces`.

Untuk informasi selengkapnya, lihat [Membandingkan wajah dalam gambar](#).

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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 uncompered = compareFacesResult.unmatchedFaces
        if (uncompered != null) {
            println("There was ${uncompered.size} face(s) that did not match")
        }

        println("Source image rotation:
        ${compareFacesResult.sourceImageOrientationCorrection}")
        println("target image rotation:
        ${compareFacesResult.targetImageOrientationCorrection}")
    }
}

```

- Untuk detail API, lihat [CompareFaces](#) di AWS SDK untuk referensi API Kotlin.



## CreateCollection

Contoh kode berikut menunjukkan cara menggunakan `CreateCollection`.

Untuk informasi selengkapnya, lihat [Membuat koleksi](#).

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [CreateCollection](#) di AWS SDK untuk referensi API Kotlin.

## DeleteCollection

Contoh kode berikut menunjukkan cara menggunakan `DeleteCollection`.

Untuk informasi selengkapnya, lihat [Menghapus koleksi](#).

## SDK untuk Kotlin

**Note**

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [DeleteCollection](#) di AWS SDK untuk referensi API Kotlin.

**DeleteFaces**

Contoh kode berikut menunjukkan cara menggunakan `DeleteFaces`.

Untuk informasi selengkapnya, lihat [Menghapus wajah dari koleksi](#).

## SDK untuk Kotlin

**Note**

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}

```

- Untuk detail API, lihat [DeleteFaces](#) di AWS SDK untuk referensi API Kotlin.

## DescribeCollection

Contoh kode berikut menunjukkan cara menggunakan `DescribeCollection`.

Untuk informasi selengkapnya, lihat [Menjelaskan koleksi](#).

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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}")
    }
}

```

- Untuk detail API, lihat [DescribeCollection](#) di AWS SDK untuk referensi API Kotlin.

## DetectFaces

Contoh kode berikut menunjukkan cara menggunakan `DetectFaces`.

Untuk informasi selengkapnya, lihat [Mendeteksi wajah dalam gambar](#).

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [DetectFaces](#) di AWS SDK untuk referensi API Kotlin.

## DetectLabels

Contoh kode berikut menunjukkan cara menggunakan `DetectLabels`.

Untuk informasi selengkapnya, lihat [Mendeteksi label dalam gambar](#).

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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


- Untuk detail API, lihat [DetectLabels](#) di AWS SDK untuk referensi API Kotlin.

## DetectModerationLabels

Contoh kode berikut menunjukkan cara menggunakan `DetectModerationLabels`.

Untuk informasi selengkapnya, lihat [Mendeteksi gambar yang tidak pantas](#).

SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [DetectModerationLabels](#) di AWS SDK untuk referensi API Kotlin.

## DetectText

Contoh kode berikut menunjukkan cara menggunakan `DetectText`.

Untuk informasi selengkapnya, lihat [Mendeteksi teks dalam gambar](#).

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [DetectText](#) di AWS SDK untuk referensi API Kotlin.

## IndexFaces

Contoh kode berikut menunjukkan cara menggunakan `IndexFaces`.

Untuk informasi selengkapnya, lihat [Menambahkan wajah ke koleksi](#).

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
        }
    }
}
```

- Untuk detail API, lihat [IndexFaces](#) di AWS SDK untuk referensi API Kotlin.

## ListCollections

Contoh kode berikut menunjukkan cara menggunakan `ListCollections`.

Untuk informasi selengkapnya, lihat [Daftar koleksi](#).

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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)
        }
    }
}
```

- Untuk detail API, lihat [ListCollections](#) di AWS SDK untuk referensi API Kotlin.

## ListFaces

Contoh kode berikut menunjukkan cara menggunakan `ListFaces`.

Untuk informasi selengkapnya, lihat [Daftar wajah dalam koleksi](#).

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListFaces](#) di AWS SDK untuk referensi API Kotlin.

## RecognizeCelebrities

Contoh kode berikut menunjukkan cara menggunakan `RecognizeCelebrities`.

Untuk informasi selengkapnya, lihat [Mengenali selebriti dalam sebuah gambar](#).

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.")
    }
}
```

- Untuk detail API, lihat [RecognizeCelebrities](#) di AWS SDK untuk referensi API Kotlin.

## Skenario

Membuat aplikasi nirserver untuk mengelola foto

Contoh kode berikut menunjukkan cara membuat aplikasi tanpa server yang memungkinkan pengguna mengelola foto menggunakan label.

SDK untuk Kotlin

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

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

Mendeteksi informasi dalam video

Contoh kode berikut ini menunjukkan cara:

- Mulai pekerjaan Amazon Rekognition untuk mendeteksi elemen seperti orang, objek, dan teks dalam video.
- Periksa status pekerjaan sampai pekerjaan selesai.
- Keluarkan daftar elemen yang terdeteksi oleh setiap pekerjaan.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

Mendeteksi wajah dalam video yang disimpan dalam bucket Amazon S3.

```
suspend fun startFaceDetection(
    channelVal: NotificationChannel?,
    bucketVal: String,
    videoVal: String,
) {
    val s3obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vid0b =
        Video {
            s3object = s3obj
        }

    val request =
        StartFaceDetectionRequest {
            jobTag = "Faces"
            faceAttributes = FaceAttributes.All
            notificationChannel = channelVal
            video = vid0b
        }

    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
```

```
RecognitionClient { 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}")
    }
}
}
```

Mendeteksi konten yang tidak pantas atau menyinggung dalam video yang disimpan di bucket Amazon S3.

```
suspend fun startModerationDetection(
    channel: NotificationChannel?,
    bucketVal: String?,
    videoVal: String?,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidObj =
        Video {
            s3Object = s3Obj
        }
    val request =
        StartContentModerationRequest {
            jobTag = "Moderation"
            notificationChannel = channel
            video = vidObj
        }

    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)
    }
}
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [GetCelebrityRecognition](#)
  - [GetContentModeration](#)
  - [GetLabelDetection](#)
  - [GetPersonTracking](#)
  - [GetSegmentDetection](#)
  - [GetTextDetection](#)
  - [StartCelebrityRecognition](#)
  - [StartContentModeration](#)
  - [StartLabelDetection](#)
  - [StartPersonTracking](#)
  - [StartSegmentDetection](#)



- [StartTextDetection](#)

Mendeteksi objek dalam gambar

Contoh kode berikut menunjukkan cara membuat aplikasi yang menggunakan Amazon Rekognition untuk mendeteksi objek berdasarkan kategori dalam gambar.

SDK untuk Kotlin

Menunjukkan cara menggunakan Amazon Rekognition Kotlin API untuk membuat aplikasi yang menggunakan Amazon Rekognition untuk mengidentifikasi objek berdasarkan kategori dalam gambar yang berada di bucket Amazon Simple Storage Service (Amazon S3). Aplikasi ini mengirimkan notifikasi email kepada admin beserta hasilnya menggunakan Amazon Simple Email Service (Amazon SES).

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Route 53 contoh pendaftaran domain menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan registrasi domain Route 53.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

## Memulai

Hello Route 53 pendaftaran domain

Contoh kode berikut menunjukkan cara memulai menggunakan pendaftaran domain Route 53.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListPrices](#) di AWS SDK untuk referensi API Kotlin.

## Topik

- [Hal-hal mendasar](#)
- [Tindakan](#)

## Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara:

- Buat daftar domain saat ini, dan daftar operasi dalam satu tahun terakhir.
- Lihat tagihan selama setahun terakhir, dan lihat harga untuk jenis domain.
- Dapatkan saran domain.
- Periksa ketersediaan domain dan transferabilitas.

- Secara opsional, minta pendaftaran domain.
- Dapatkan detail operasi.
- Secara opsional, dapatkan detail domain.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
/**
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 xxxxxxxx not found in xxxxxxxx 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}")
            }
    }
}
}
```


- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [CheckDomainAvailability](#)
  - [CheckDomainTransferability](#)
  - [GetDomainDetail](#)
  - [GetDomainSuggestions](#)
  - [GetOperationDetail](#)
  - [ListDomains](#)
  - [ListOperations](#)
  - [ListPrices](#)
  - [RegisterDomain](#)
  - [ViewBilling](#)

## Tindakan

### **CheckDomainAvailability**

Contoh kode berikut menunjukkan cara menggunakan `CheckDomainAvailability`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [CheckDomainAvailability](#) di AWS SDK untuk referensi API Kotlin.

## CheckDomainTransferability

Contoh kode berikut menunjukkan cara menggunakan `CheckDomainTransferability`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}

```

- Untuk detail API, lihat [CheckDomainTransferability](#) di AWS SDK untuk referensi API Kotlin.

## GetDomainDetail

Contoh kode berikut menunjukkan cara menggunakan `GetDomainDetail`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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}")
    }
}

```

- Untuk detail API, lihat [GetDomainDetail](#) di AWS SDK untuk referensi API Kotlin.

## GetDomainSuggestions

Contoh kode berikut menunjukkan cara menggunakan `GetDomainSuggestions`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [GetDomainSuggestions](#) di AWS SDK untuk referensi API Kotlin.

## GetOperationDetail

Contoh kode berikut menunjukkan cara menggunakan `GetOperationDetail`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [GetOperationDetail](#) di AWS SDK untuk referensi API Kotlin.

## ListDomains

Contoh kode berikut menunjukkan cara menggunakan `ListDomains`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
}

```

- Untuk detail API, lihat [ListDomains](#) di AWS SDK untuk referensi API Kotlin.

## ListOperations

Contoh kode berikut menunjukkan cara menggunakan `ListOperations`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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}")
            }
    }
}

```



```
    }
}
```

- Untuk detail API, lihat [ListOperations](#) di AWS SDK untuk referensi API Kotlin.

## ListPrices

Contoh kode berikut menunjukkan cara menggunakan `ListPrices`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListPrices](#) di AWS SDK untuk referensi API Kotlin.

## RegisterDomain

Contoh kode berikut menunjukkan cara menggunakan `RegisterDomain`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}

```

- Untuk detail API, lihat [RegisterDomain](#) di AWS SDK untuk referensi API Kotlin.

## ViewBilling

Contoh kode berikut menunjukkan cara menggunakan `ViewBilling`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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}")
            }
    }
}
```

- Untuk detail API, lihat [ViewBilling](#) di AWS SDK untuk referensi API Kotlin.

## Contoh Amazon S3 menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Amazon S3.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Hal-hal mendasar](#)
- [Tindakan](#)
- [Skenario](#)

## Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara:

- Membuat bucket dan mengunggah file ke dalamnya.
- Mengunduh objek dari bucket.
- Menyalin objek ke subfolder di bucket.
- Membuat daftar objek dalam bucket.
- Menghapus objek bucket dan bucket tersebut.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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 delObj =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delObj
        }

    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!")
    }
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjectsV2](#)
  - [PutObject](#)

## Tindakan

### CopyObject

Contoh kode berikut menunjukkan cara menggunakan CopyObject.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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)
    }
}
```

- Untuk detail API, lihat [CopyObject](#) di AWS SDK untuk referensi API Kotlin.

## CreateBucket

Contoh kode berikut menunjukkan cara menggunakan `CreateBucket`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun createNewBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}
```

- Untuk detail API, lihat [CreateBucket](#) di AWS SDK untuk referensi API Kotlin.

## CreateMultiRegionAccessPoint

Contoh kode berikut menunjukkan cara menggunakan `CreateMultiRegionAccessPoint`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

Konfigurasi klien kontrol S3 untuk mengirim permintaan ke Wilayah us-west-2.

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

### Buat Titik Akses Multi-Wilayah.

```
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"
}

```

Tunggu hingga Titik Akses Multi-Wilayah tersedia.

```

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)
    }
}

```

- Untuk informasi selengkapnya, lihat [AWS panduan pengembang SDK untuk Kotlin](#).
- Untuk detail API, lihat [CreateMultiRegionAccessPoint](#) di AWS SDK untuk referensi API Kotlin.

## DeleteBucketPolicy

Contoh kode berikut menunjukkan cara menggunakan `DeleteBucketPolicy`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- Untuk detail API, lihat [DeleteBucketPolicy](#) di AWS SDK untuk referensi API Kotlin.

## DeleteObjects

Contoh kode berikut menunjukkan cara menggunakan `DeleteObjects`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [DeleteObjects](#) di AWS SDK untuk referensi API Kotlin.

## GetBucketPolicy

Contoh kode berikut menunjukkan cara menggunakan `GetBucketPolicy`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}
```

- Untuk detail API, lihat [GetBucketPolicy](#) di AWS SDK untuk referensi API Kotlin.

## GetObject

Contoh kode berikut menunjukkan cara menggunakan `GetObject`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
        }
    }
}
```

- Untuk detail API, lihat [GetObject](#) di AWS SDK untuk referensi API Kotlin.

## GetObjectAcl

Contoh kode berikut menunjukkan cara menggunakan `GetObjectAcl`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
        }
    }
}

```

- Untuk detail API, lihat [GetObjectAcl](#) di AWS SDK untuk referensi API Kotlin.

## ListObjectsV2

Contoh kode berikut menunjukkan cara menggunakan `ListObjectsV2`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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

```

- Untuk detail API, lihat [ListObjectsV2](#) di AWS SDK untuk referensi API Kotlin.

## PutBucketAcl

Contoh kode berikut menunjukkan cara menggunakan `PutBucketAcl`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}

```

- Untuk detail API, lihat [PutBucketAcl](#) di AWS SDK untuk referensi API Kotlin.

## PutObject

Contoh kode berikut menunjukkan cara menggunakan `PutObject`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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}")
    }
}
```

- Untuk detail API, lihat [PutObject](#) di AWS SDK untuk referensi API Kotlin.

## Skenario

Membuat URL yang telah ditetapkan sebelumnya

Contoh kode berikut menunjukkan cara membuat URL presigned untuk Amazon S3 dan mengunggah objek.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat permintaan `GetObject` yang telah ditetapkan sebelumnya dan gunakan URL untuk mengunduh objek.

```
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
}

```

Buat permintaan yang `GetObject` telah ditetapkan sebelumnya dengan opsi lanjutan.

```

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
}

```

Buat permintaan `PutObject` yang telah ditetapkan sebelumnya dan gunakan untuk mengunggah objek.

```
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)
}
```

- Untuk informasi selengkapnya, lihat [AWS panduan pengembang SDK untuk Kotlin](#).

## Membuat aplikasi nirserver untuk mengelola foto

Contoh kode berikut menunjukkan cara membuat aplikasi tanpa server yang memungkinkan pengguna mengelola foto menggunakan label.

## SDK untuk Kotlin

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

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

## Mendeteksi objek dalam gambar

Contoh kode berikut menunjukkan cara membuat aplikasi yang menggunakan Amazon Rekognition untuk mendeteksi objek berdasarkan kategori dalam gambar.

## SDK untuk Kotlin

Menunjukkan cara menggunakan Amazon Rekognition Kotlin API untuk membuat aplikasi yang menggunakan Amazon Rekognition untuk mengidentifikasi objek berdasarkan kategori dalam gambar yang berada di bucket Amazon Simple Storage Service (Amazon S3). Aplikasi ini mengirimkan notifikasi email kepada admin beserta hasilnya menggunakan Amazon Simple Email Service (Amazon SES).

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3
- Amazon SES



## Dapatkan objek dari Titik Akses Multi-Region

Contoh kode berikut menunjukkan bagaimana untuk mendapatkan objek dari Multi-Region Access Point.

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

Konfigurasi klien S3 untuk menggunakan algoritma penandatanganan Asymmetric Sigv4 (Sigv4a).

```
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
}
```

Gunakan ARN Titik Akses Multi-Wilayah sebagai ganti nama bucket untuk mengambil objek.

```
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 $mrpArn")
        }
    }
    return stringObj
}
```

- Untuk informasi selengkapnya, lihat [AWS panduan pengembang SDK untuk Kotlin](#).
- Untuk detail API, lihat [GetObject](#) di AWS SDK untuk referensi API Kotlin.

## SageMaker Contoh AI menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan SageMaker AI.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Memulai

Halo SageMaker AI

Contoh kode berikut menunjukkan cara memulai menggunakan SageMaker AI.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh selengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListNotebookInstances](#) di AWS SDK untuk referensi API Kotlin.

## Topik

- [Tindakan](#)
- [Skenario](#)

## Tindakan

### CreatePipeline

Contoh kode berikut menunjukkan cara menggunakan CreatePipeline.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
// 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)
}
}
}

```

- Untuk detail API, lihat [CreatePipeline](#) di AWS SDK untuk referensi API Kotlin.

## DeletePipeline

Contoh kode berikut menunjukkan cara menggunakan `DeletePipeline`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

// 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")
}
}

```

- Untuk detail API, lihat [DeletePipeline](#) di AWS SDK untuk referensi API Kotlin.

## DescribePipelineExecution

Contoh kode berikut menunjukkan cara menggunakan `DescribePipelineExecution`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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")
}
```

- Untuk detail API, lihat [DescribePipelineExecution](#) di AWS SDK untuk referensi API Kotlin.

## StartPipelineExecution

Contoh kode berikut menunjukkan cara menggunakan `StartPipelineExecution`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
// 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
}
}
```

- Untuk detail API, lihat [StartPipelineExecution](#) di AWS SDK untuk referensi API Kotlin.

## Skenario

Memulai pekerjaan geospasial dan jaringan pipa

Contoh kode berikut ini menunjukkan cara:

- Siapkan sumber daya untuk pipa.
- Siapkan pipa yang menjalankan pekerjaan geospasial.
- Mulai eksekusi pipeline.
- Pantau status eksekusi.
- Lihat output dari pipa.
- Pembersihan sumber daya

Untuk informasi selengkapnya, lihat [Membuat dan menjalankan SageMaker pipeline menggunakan AWS SDKs Community.aws](#).



## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

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.waitForFunctionActive {
            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
}
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [CreatePipeline](#)
  - [DeletePipeline](#)
  - [DescribePipelineExecution](#)
  - [StartPipelineExecution](#)
  - [UpdatePipeline](#)

## Contoh Secrets Manager menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Kotlin with Secrets Manager.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Tindakan](#)

## Tindakan

### GetSecretValue

Contoh kode berikut menunjukkan cara menggunakan `GetSecretValue`.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
}
}
```

- Untuk detail API, lihat [GetSecretValue](#) di AWS SDK untuk referensi API Kotlin.

## Contoh Amazon SES menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Amazon SES.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- [Skenario](#)

## Skenario

Membuat aplikasi web untuk melacak data DynamoDB

Contoh kode berikut menunjukkan cara membuat aplikasi web yang melacak item kerja dalam tabel Amazon DynamoDB dan menggunakan Amazon Simple Email Service (Amazon SES) untuk mengirim laporan.

## SDK untuk Kotlin

Menunjukkan cara menggunakan Amazon DynamoDB API untuk membuat aplikasi web dinamis yang melacak data kerja DynamoDB.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- DynamoDB
- Amazon SES

## Buat aplikasi web untuk melacak data Amazon Redshift

Contoh kode berikut menunjukkan cara membuat aplikasi web yang melacak dan melaporkan item pekerjaan menggunakan database Amazon Redshift.

## SDK untuk Kotlin

Menunjukkan cara membuat aplikasi web yang melacak dan melaporkan item pekerjaan yang disimpan dalam database Amazon Redshift.

Untuk kode sumber lengkap dan petunjuk tentang cara menyiapkan Spring REST API yang menanyakan data Amazon Redshift dan untuk digunakan oleh aplikasi React, lihat contoh lengkapnya di [GitHub](#)

Layanan yang digunakan dalam contoh ini

- Amazon Redshift
- Amazon SES

## Buat pelacak butir kerja Aurora Nirserver

Contoh kode berikut menunjukkan cara membuat aplikasi web yang melacak item pekerjaan dalam database Amazon Aurora Tanpa Server dan menggunakan Amazon Simple Email Service (Amazon SES) untuk mengirim laporan.

## SDK untuk Kotlin

Menunjukkan cara membuat aplikasi web yang melacak dan melaporkan butir kerja yang tersimpan dalam basis data Amazon RDS.

Untuk kode sumber lengkap dan petunjuk tentang cara menyiapkan Spring REST API yang menanyakan data Amazon Aurora Tanpa Server dan untuk digunakan oleh aplikasi React, lihat contoh lengkapnya di [GitHub](#)

Layanan yang digunakan dalam contoh ini

- Aurora
- Amazon RDS
- Layanan Data Amazon RDS
- Amazon SES

Mendeteksi objek dalam gambar

Contoh kode berikut menunjukkan cara membuat aplikasi yang menggunakan Amazon Rekognition untuk mendeteksi objek berdasarkan kategori dalam gambar.

SDK untuk Kotlin

Menunjukkan cara menggunakan Amazon Rekognition Kotlin API untuk membuat aplikasi yang menggunakan Amazon Rekognition untuk mengidentifikasi objek berdasarkan kategori dalam gambar yang berada di bucket Amazon Simple Storage Service (Amazon S3). Aplikasi ini mengirimkan notifikasi email kepada admin beserta hasilnya menggunakan Amazon Simple Email Service (Amazon SES).

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Contoh Amazon SNS menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Amazon SNS.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.


Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Memulai

Halo Amazon SNS

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon SNS.

SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListTopics](#) di AWS SDK untuk referensi API Kotlin.

## Topik

- [Tindakan](#)
- [Skenario](#)

## Tindakan

### CreateTopic

Contoh kode berikut menunjukkan cara menggunakan `CreateTopic`.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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()
    }
}
```

- Untuk detail API, lihat [CreateTopic](#) di AWS SDK untuk referensi API Kotlin.

## DeleteTopic

Contoh kode berikut menunjukkan cara menggunakan `DeleteTopic`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.")
    }
}
```

- Untuk detail API, lihat [DeleteTopic](#) di AWS SDK untuk referensi API Kotlin.

## GetTopicAttributes

Contoh kode berikut menunjukkan cara menggunakan `GetTopicAttributes`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [GetTopicAttributes](#) di AWS SDK untuk referensi API Kotlin.

## ListSubscriptions

Contoh kode berikut menunjukkan cara menggunakan `ListSubscriptions`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListSubscriptions](#) di AWS SDK untuk referensi API Kotlin.

## ListTopics

Contoh kode berikut menunjukkan cara menggunakan `ListTopics`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).


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

- Untuk detail API, lihat [ListTopics](#) di AWS SDK untuk referensi API Kotlin.

## Publish

Contoh kode berikut menunjukkan cara menggunakan `Publish`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.")
    }
}
```

- Untuk detail API, lihat [Menerbitkan](#) di AWS SDK untuk referensi API Kotlin.

## SetTopicAttributes

Contoh kode berikut menunjukkan cara menggunakan `SetTopicAttributes`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.")
    }
}
```

- Untuk detail API, lihat [SetTopicAttributes](#) di AWS SDK untuk referensi API Kotlin.

## Subscribe

Contoh kode berikut menunjukkan cara menggunakan `Subscribe`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

Berlangganan alamat email ke suatu topik.

```
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()
    }
}
```

Berlangganan fungsi Lambda ke suatu topik.

```
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}")
    }
}
```

- Untuk detail API, lihat [Berlangganan](#) di AWS SDK untuk referensi API Kotlin.

## TagResource

Contoh kode berikut menunjukkan cara menggunakan TagResource.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [TagResource](#) di AWS SDK untuk referensi API Kotlin.

## Unsubscribe

Contoh kode berikut menunjukkan cara menggunakan `Unsubscribe`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [Berhenti berlangganan](#) di AWS SDK untuk referensi API Kotlin.

## Skenario

Membangun aplikasi untuk mengirimkan data ke tabel DynamoDB

Contoh kode berikut menunjukkan cara membuat aplikasi yang mengirimkan data ke tabel Amazon DynamoDB dan memberi tahu Anda saat pengguna memperbarui tabel.

SDK untuk Kotlin

Menunjukkan cara membuat aplikasi Android native yang mengirimkan data menggunakan API Kotlin Amazon DynamoDB dan mengirim pesan teks menggunakan API Kotlin Amazon SNS.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).



Layanan yang digunakan dalam contoh ini

- DynamoDB
- Amazon SNS

## Membangun aplikasi Amazon SNS

Contoh kode berikut menunjukkan cara membuat aplikasi yang memiliki langganan dan mempublikasikan fungsionalitas dan menerjemahkan pesan.

### SDK untuk Kotlin

Menunjukkan cara menggunakan Amazon SNS Kotlin API untuk membuat aplikasi yang memiliki fungsionalitas langganan dan publikasi. Selain itu, contoh aplikasi ini juga menerjemahkan pesan.

Untuk kode sumber lengkap dan petunjuk tentang cara membuat aplikasi web, lihat contoh lengkapnya di [GitHub](#).

Untuk kode sumber lengkap dan petunjuk tentang cara membuat aplikasi Android asli, lihat contoh selengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon SNS
- Amazon Translate

## Membuat aplikasi nirserver untuk mengelola foto

Contoh kode berikut menunjukkan cara membuat aplikasi tanpa server yang memungkinkan pengguna mengelola foto menggunakan label.

### SDK untuk Kotlin

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).


Layanan yang digunakan dalam contoh ini

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

Publikasikan pesan teks SMS

Contoh kode berikut menunjukkan cara mempublikasikan pesan SMS menggunakan Amazon SNS.

SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.")
    }
}
```

- Untuk detail API, lihat [Menerbitkan](#) di AWS SDK untuk referensi API Kotlin.

## Publikasikan pesan ke antrian

Contoh kode berikut ini menunjukkan cara:

- Buat topik (FIFO atau non-FIFO).
- Berlangganan beberapa antrian ke topik dengan opsi untuk menerapkan filter.
- Publikasikan pesan ke topik.
- Polling antrian untuk pesan yang diterima.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [CreateQueue](#)
  - [CreateTopic](#)
  - [DeleteMessageBatch](#)
  - [DeleteQueue](#)
  - [DeleteTopic](#)
  - [GetQueueAttributes](#)
  - [Publikasikan](#)
  - [ReceiveMessage](#)
  - [SetQueueAttributes](#)
  - [Berlangganan](#)
  - [Berhenti berlangganan](#)

## Contoh Amazon SQS menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Amazon SQS.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Memulai

Halo Amazon SQS

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon SQS.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListQueues](#) di AWS SDK untuk referensi API Kotlin.

## Topik

- [Tindakan](#)
- [Skenario](#)

## Tindakan

### CreateQueue

Contoh kode berikut menunjukkan cara menggunakan `CreateQueue`.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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()
}
}
```

- Untuk detail API, lihat [CreateQueue](#) di AWS SDK untuk referensi API Kotlin.

## DeleteMessage

Contoh kode berikut menunjukkan cara menggunakan `DeleteMessage`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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!")
    }
}
```

- Untuk detail API, lihat [DeleteMessage](#) di AWS SDK untuk referensi API Kotlin.

## DeleteQueue

Contoh kode berikut menunjukkan cara menggunakan `DeleteQueue`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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!")
    }
}
```

- Untuk detail API, lihat [DeleteQueue](#) di AWS SDK untuk referensi API Kotlin.

## ListQueues

Contoh kode berikut menunjukkan cara menggunakan `ListQueues`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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)
        }
    }
}
```

```
}
```

- Untuk detail API, lihat [ListQueues](#) di AWS SDK untuk referensi API Kotlin.

## ReceiveMessage

Contoh kode berikut menunjukkan cara menggunakan `ReceiveMessage`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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)
        }
    }
}
```

- Untuk detail API, lihat [ReceiveMessage](#) di AWS SDK untuk referensi API Kotlin.

## SendMessage

Contoh kode berikut menunjukkan cara menggunakan `SendMessage`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.")
}
}
```

- Untuk detail API, lihat [SendMessage](#) di AWS SDK untuk referensi API Kotlin.

## Skenario

Buat aplikasi perpesanan

Contoh kode berikut menunjukkan cara membuat aplikasi pesan dengan menggunakan Amazon SQS.

SDK untuk Kotlin

Menunjukkan cara menggunakan Amazon SQS API untuk mengembangkan Spring REST API yang mengirim dan mengambil pesan.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Comprehend
- Amazon SQS

Publikasikan pesan ke antrian

Contoh kode berikut ini menunjukkan cara:

- Buat topik (FIFO atau non-FIFO).
- Berlangganan beberapa antrian ke topik dengan opsi untuk menerapkan filter.

- Publikasikan pesan ke topik.
- Polling antrian untuk pesan yang diterima.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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": {
```



```
        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
    }
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [CreateQueue](#)
  - [CreateTopic](#)
  - [DeleteMessageBatch](#)
  - [DeleteQueue](#)
  - [DeleteTopic](#)
  - [GetQueueAttributes](#)
  - [Publikasikan](#)
  - [ReceiveMessage](#)
  - [SetQueueAttributes](#)
  - [Berlangganan](#)
  - [Berhenti berlangganan](#)

## Contoh Step Functions menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Step Functions.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.



Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.


Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Memulai

Hello Step Functions

Contoh kode berikut menunjukkan cara memulai menggunakan Step Functions.

SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [ListStateMachines](#) di AWS SDK untuk referensi API Kotlin.

## Topik

- [Hal-hal mendasar](#)
- [Tindakan](#)

## Hal-hal mendasar

### Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara:

- Buat aktivitas.
- Buat mesin status dari definisi Bahasa Negara Amazon yang berisi aktivitas yang dibuat sebelumnya sebagai langkah.
- Jalankan mesin status dan tanggapilah aktivitas dengan input pengguna.
- Dapatkan status dan output akhir setelah proses selesai, lalu bersihkan sumber daya.

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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](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
    }
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [CreateActivity](#)
  - [CreateStateMachine](#)
  - [DeleteActivity](#)
  - [DeleteStateMachine](#)
  - [DescribeExecution](#)
  - [DescribeStateMachine](#)
  - [GetActivityTask](#)
  - [ListActivities](#)
  - [ListStateMachines](#)
  - [SendTaskSuccess](#)
  - [StartExecution](#)

- [StopExecution](#)

## Tindakan

### CreateActivity

Contoh kode berikut menunjukkan cara menggunakan `CreateActivity`.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}
```

- Untuk detail API, lihat [CreateActivity](#) di AWS SDK untuk referensi API Kotlin.

### CreateStateMachine

Contoh kode berikut menunjukkan cara menggunakan `CreateStateMachine`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}
```

- Untuk detail API, lihat [CreateStateMachine](#) di AWS SDK untuk referensi API Kotlin.

## DeleteActivity

Contoh kode berikut menunjukkan cara menggunakan `DeleteActivity`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
    }
}
```

- Untuk detail API, lihat [DeleteActivity](#) di AWS SDK untuk referensi API Kotlin.

## DeleteStateMachine

Contoh kode berikut menunjukkan cara menggunakan `DeleteStateMachine`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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.")
    }
}
```

- Untuk detail API, lihat [DeleteStateMachine](#) di AWS SDK untuk referensi API Kotlin.

## DescribeExecution

Contoh kode berikut menunjukkan cara menggunakan `DescribeExecution`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
}
```

- Untuk detail API, lihat [DescribeExecution](#) di AWS SDK untuk referensi API Kotlin.

## DescribeStateMachine

Contoh kode berikut menunjukkan cara menggunakan `DescribeStateMachine`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [DescribeStateMachine](#) di AWS SDK untuk referensi API Kotlin.

## GetActivityTask

Contoh kode berikut menunjukkan cara menggunakan `GetActivityTask`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).



```
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
    }
}
```

- Untuk detail API, lihat [GetActivityTask](#) di AWS SDK untuk referensi API Kotlin.

## ListActivities

Contoh kode berikut menunjukkan cara menggunakan `ListActivities`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

```
}  
}
```

- Untuk detail API, lihat [ListActivities](#) di AWS SDK untuk referensi API Kotlin.

## ListExecutions

Contoh kode berikut menunjukkan cara menggunakan `ListExecutions`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListExecutions](#) di AWS SDK untuk referensi API Kotlin.

## ListStateMachines

Contoh kode berikut menunjukkan cara menggunakan `ListStateMachines`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [ListStateMachines](#) di AWS SDK untuk referensi API Kotlin.

## SendTaskSuccess

Contoh kode berikut menunjukkan cara menggunakan `SendTaskSuccess`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
            output = json
        }
    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}
```

- Untuk detail API, lihat [SendTaskSuccess](#) di AWS SDK untuk referensi API Kotlin.

## StartExecution

Contoh kode berikut menunjukkan cara menggunakan `StartExecution`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}
```

- Untuk detail API, lihat [StartExecution](#) di AWS SDK untuk referensi API Kotlin.

## Dukungan contoh menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK untuk Kotlin with. Dukungan

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Memulai

Halo Dukungan

Contoh kode berikut menunjukkan cara untuk mulai menggunakan Dukungan.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
/**
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++
    }
}
}
```

- Untuk detail API, lihat [DescribeServices](#) di AWS SDK untuk referensi API Kotlin.

## Topik

- [Hal-hal mendasar](#)
- [Tindakan](#)


## Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara:

- Dapatkan dan tampilkan layanan yang tersedia dan tingkat keparahan untuk kasus.
- Buat kasus dukungan menggunakan layanan, kategori, dan tingkat keparahan yang dipilih.
- Dapatkan dan tampilkan daftar kasus terbuka untuk hari ini.
- Tambahkan set lampiran dan komunikasi ke kasus baru.
- Jelaskan keterikatan dan komunikasi baru untuk kasus ini.
- Selesaikan kasusnya.
- Dapatkan dan tampilkan daftar kasus yang diselesaikan untuk hari ini.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
/**
```

```
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  
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
  - [AddAttachmentsToSet](#)
  - [AddCommunicationToCase](#)
  - [CreateCase](#)
  - [DescribeAttachment](#)
  - [DescribeCases](#)
  - [DescribeCommunications](#)
  - [DescribeServices](#)
  - [DescribeSeverityLevels](#)
  - [ResolveCase](#)

## Tindakan

### AddAttachmentsToSet

Contoh kode berikut menunjukkan cara menggunakanAddAttachmentsToSet.

SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}
```

- Untuk detail API, lihat [AddAttachmentsToSet](#) di AWS SDK untuk referensi API Kotlin.

## AddCommunicationToCase

Contoh kode berikut menunjukkan cara menggunakan `AddCommunicationToCase`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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")
        }
    }
}
```

- Untuk detail API, lihat [AddCommunicationToCase](#) di AWS SDK untuk referensi API Kotlin.

## CreateCase

Contoh kode berikut menunjukkan cara menggunakan `CreateCase`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
}
}
```

- Untuk detail API, lihat [CreateCase](#) di AWS SDK untuk referensi API Kotlin.

## DescribeAttachment

Contoh kode berikut menunjukkan cara menggunakan `DescribeAttachment`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [DescribeAttachment](#) di AWS SDK untuk referensi API Kotlin.

## DescribeCases

Contoh kode berikut menunjukkan cara menggunakan `DescribeCases`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

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

- Untuk detail API, lihat [DescribeCases](#) di AWS SDK untuk referensi API Kotlin.

## DescribeCommunications

Contoh kode berikut menunjukkan cara menggunakan `DescribeCommunications`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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 ""
}
```

- Untuk detail API, lihat [DescribeCommunications](#) di AWS SDK untuk referensi API Kotlin.

## DescribeServices

Contoh kode berikut menunjukkan cara menggunakan `DescribeServices`.

## SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
// 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
    }
}
```

- Untuk detail API, lihat [DescribeServices](#) di AWS SDK untuk referensi API Kotlin.

## DescribeSeverityLevels

Contoh kode berikut menunjukkan cara menggunakan `DescribeSeverityLevels`.

SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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
    }
}
```

- Untuk detail API, lihat [DescribeSeverityLevels](#) di AWS SDK untuk referensi API Kotlin.

## ResolveCase

Contoh kode berikut menunjukkan cara menggunakan `ResolveCase`.

## SDK untuk Kotlin

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturannya dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
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}")
    }
}
```

- Untuk detail API, lihat [ResolveCase](#) di AWS SDK untuk referensi API Kotlin.

## Contoh Amazon Translate menggunakan SDK untuk Kotlin

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK untuk Kotlin dengan Amazon Translate.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

### Topik

- [Skenario](#)

## Skenario

### Membangun aplikasi Amazon SNS

Contoh kode berikut menunjukkan cara membuat aplikasi yang memiliki langganan dan mempublikasikan fungsionalitas dan menerjemahkan pesan.

#### SDK untuk Kotlin

Menunjukkan cara menggunakan Amazon SNS Kotlin API untuk membuat aplikasi yang memiliki fungsionalitas langganan dan publikasi. Selain itu, contoh aplikasi ini juga menerjemahkan pesan.

Untuk kode sumber lengkap dan petunjuk tentang cara membuat aplikasi web, lihat contoh lengkapnya di [GitHub](#).

Untuk kode sumber lengkap dan petunjuk tentang cara membuat aplikasi Android asli, lihat contoh selengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon SNS
- Amazon Translate



# Keamanan untuk AWS SDK for Kotlin

Keamanan cloud di Amazon Web Services (AWS) merupakan prioritas tertinggi. Sebagai seorang pelanggan AWS, Anda mendapatkan manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan dari organisasi yang paling sensitif terhadap keamanan. Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. [Model Tanggung Jawab Bersama](#) menggambarkan ini sebagai Keamanan dari Cloud dan Keamanan dalam Cloud.

Security of the Cloud - AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan semua layanan yang ditawarkan di AWS Cloud dan memberi Anda layanan yang dapat Anda gunakan dengan aman. Tanggung jawab keamanan kami adalah prioritas tertinggi di AWS, dan efektivitas keamanan kami secara teratur diuji dan diverifikasi oleh auditor pihak ketiga sebagai bagian dari [Program AWS Kepatuhan](#).

Keamanan di Cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan, dan faktor-faktor lain termasuk sensitivitas data Anda, persyaratan organisasi Anda, serta undang-undang dan peraturan yang berlaku.

AWS Produk atau layanan ini mengikuti [model tanggung jawab bersama](#) melalui layanan Amazon Web Services (AWS) tertentu yang didukungnya. Untuk informasi keamanan AWS layanan, lihat [halaman dokumentasi keamanan AWS layanan](#) dan [AWS layanan yang berada dalam lingkup upaya AWS kepatuhan oleh program kepatuhan](#).

## Topik

- [Perlindungan data di AWS SDK for Kotlin](#)
- [AWS SDK for Kotlin dukungan untuk TLS 1.2](#)
- [Identity and Access Management](#)
- [Validasi Kepatuhan untuk AWS Produk atau Layanan ini](#)
- [Ketahanan untuk AWS Produk atau Layanan ini](#)
- [Keamanan Infrastruktur untuk AWS Produk atau Layanan ini](#)

## Perlindungan data di AWS SDK for Kotlin

[Model tanggung jawab AWS bersama model](#) berlaku untuk perlindungan data di AWS SDK for Kotlin. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur

global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam [Pertanyaan Umum Privasi Data](#). Lihat informasi tentang perlindungan data di Eropa di pos blog [Model Tanggung Jawab Bersama dan GDPR AWS](#) di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensial dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya. AWS Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan logging aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat [Bekerja dengan CloudTrail jejak](#) di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-3 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi selengkapnya tentang titik akhir FIPS yang tersedia di [Standar Pemrosesan Informasi Federal \(FIPS\) 140-3](#).

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan SDK untuk Kotlin atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau. AWS SDKs Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

## AWS SDK for Kotlin dukungan untuk TLS 1.2

Informasi berikut hanya berlaku untuk implementasi Java SSL (implementasi SSL default dalam AWS SDK for Kotlin penargetan JVM). Jika Anda menggunakan implementasi SSL yang berbeda, lihat implementasi SSL spesifik Anda untuk mempelajari cara menerapkan versi TLS.

### Dukungan TLS di Jawa

TLS 1.2 didukung mulai dari Java 7.

### Cara memeriksa versi TLS

Untuk memeriksa versi TLS apa yang didukung di mesin virtual Java Anda (JVM), Anda dapat menggunakan kode berikut.

```
println(SSLContext.getDefault().supportedSSLParameters.protocols.joinToString(separator = ", "))
```

Untuk melihat jabat tangan SSL beraksi dan versi TLS apa yang digunakan, Anda dapat menggunakan properti sistem `javax.net.debug`.

```
-Djavax.net.debug=ssl
```

## Identity and Access Management

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya. AWS IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- [Audiens](#)
- [Mengautentikasi dengan identitas](#)
- [Mengelola akses menggunakan kebijakan](#)
- [Bagaimana Layanan AWS bekerja dengan IAM](#)

- [Memecahkan masalah AWS identitas dan akses](#)

## Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan. AWS

**Pengguna layanan** — Jika Anda menggunakan Layanan AWS untuk melakukan pekerjaan Anda, maka administrator Anda memberi Anda kredensial dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak AWS fitur untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur AWS, lihat [Memecahkan masalah AWS identitas dan akses](#) atau panduan pengguna yang Layanan AWS Anda gunakan.

**Administrator layanan** — Jika Anda bertanggung jawab atas AWS sumber daya di perusahaan Anda, Anda mungkin memiliki akses penuh ke AWS. Tugas Anda adalah menentukan AWS fitur dan sumber daya mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep dasar IAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM AWS, lihat panduan pengguna yang Layanan AWS Anda gunakan.

**Administrator IAM** – Jika Anda adalah administrator IAM, Anda mungkin ingin belajar dengan lebih detail tentang cara Anda menulis kebijakan untuk mengelola akses ke AWS. Untuk melihat contoh kebijakan AWS berbasis identitas yang dapat Anda gunakan di IAM, lihat panduan pengguna yang Anda gunakan. Layanan AWS

## Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensial yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensial Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas terfederasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat [Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS](#).

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensial Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Guna mengetahui informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat [AWS Signature Version 4 untuk permintaan API](#) dalam Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, lihat [Autentikasi multi-faktor](#) dalam Panduan Pengguna AWS IAM Identity Center dan [Autentikasi multi-faktor AWS di IAM](#) dalam Panduan Pengguna IAM.

## Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat [Tugas yang memerlukan kredensial pengguna root](#) dalam Panduan Pengguna IAM.

## Identitas gabungan

Sebagai praktik terbaik, mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS dengan menggunakan kredensial sementara.

Identitas federasi adalah pengguna dari direktori pengguna perusahaan Anda, penyedia identitas web, direktori Pusat Identitas AWS Directory Service, atau pengguna mana pun yang mengakses Layanan AWS dengan menggunakan kredensial yang disediakan melalui sumber identitas. Ketika identitas federasi mengakses Akun AWS, mereka mengambil peran, dan peran memberikan kredensial sementara.

Untuk manajemen akses terpusat, kami sarankan Anda menggunakan AWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat Identitas IAM, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan di semua aplikasi Akun AWS dan aplikasi Anda. Untuk informasi tentang Pusat Identitas IAM, lihat [Apakah itu Pusat Identitas IAM?](#) dalam Panduan Pengguna AWS IAM Identity Center .

## Pengguna dan grup IAM

[Pengguna IAM](#) adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, kami merekomendasikan untuk mengandalkan kredensial sementara, bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan tertentu yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami merekomendasikan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat [Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang](#) dalam Panduan Pengguna IAM.

[Grup IAM](#) adalah identitas yang menentukan sekumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat meminta kelompok untuk menyebutkan IAMAdmins dan memberikan izin kepada grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, lihat [Kasus penggunaan untuk pengguna IAM](#) dalam Panduan Pengguna IAM.

## Peran IAM

[Peran IAM](#) adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Untuk mengambil peran IAM sementara AWS Management Console, Anda dapat [beralih dari pengguna ke peran IAM \(konsol\)](#). Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang cara menggunakan peran, lihat [Metode untuk mengambil peran](#) dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat [Buat peran untuk penyedia identitas pihak ketiga](#) dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM akan mengorelasikan set izin ke peran dalam IAM. Untuk informasi tentang set izin, lihat [Set izin](#) dalam Panduan Pengguna AWS IAM Identity Center .
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (prinsipal tepercaya) di akun lain untuk mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Akses sumber daya lintas akun di IAM](#) dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Misalnya, saat Anda melakukan panggilan dalam suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.
  - Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat [Sesi akses maju](#).
- Peran layanan – Peran layanan adalah [peran IAM](#) yang dijalankan oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat [Buat sebuah peran untuk mendelegasikan izin ke Layanan AWS](#) dalam Panduan pengguna IAM.



- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan peran IAM untuk mengelola kredensi sementara untuk aplikasi yang berjalan pada EC2 instance dan membuat AWS CLI atau AWS permintaan API. Ini lebih baik untuk menyimpan kunci akses dalam EC2 instance. Untuk menetapkan AWS peran ke EC2 instance dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instance berisi peran dan memungkinkan program yang berjalan pada EC2 instance untuk mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat [Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di EC2 instans Amazon di Panduan Pengguna IAM](#).

## Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat [Gambaran umum kebijakan JSON](#) dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasinya. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut bisa mendapatkan informasi peran dari AWS Management Console, API AWS CLI, atau AWS API.



## Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat [Tentukan izin IAM kustom dengan kebijakan terkelola pelanggan](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat dilampirkan ke beberapa pengguna, grup, dan peran dalam akun AWS. Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan yang dikelola atau kebijakan inline, lihat [Pilih antara kebijakan yang dikelola dan kebijakan inline](#) dalam Panduan Pengguna IAM.

## Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus [menentukan prinsipal](#) dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau Layanan AWS.

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

## Daftar kontrol akses (ACLs)

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLs mirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACLs. Untuk mempelajari selengkapnya ACLs, lihat [Ringkasan daftar kontrol akses \(ACL\)](#) di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

## Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- **Batasan izin** – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas ke entitas IAM (pengguna IAM atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batasan izin, lihat [Batasan izin untuk entitas IAM](#) dalam Panduan Pengguna IAM.
- **Kebijakan kontrol layanan (SCPs)** — SCPs adalah kebijakan JSON yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur dalam suatu organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCPs) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS Untuk informasi selengkapnya tentang Organizations dan SCPs, lihat [Kebijakan kontrol layanan](#) di Panduan AWS Organizations Pengguna.
- **Kebijakan kontrol sumber daya (RCPs)** — RCPs adalah kebijakan JSON yang dapat Anda gunakan untuk menetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda tanpa memperbarui kebijakan IAM yang dilampirkan ke setiap sumber daya yang Anda miliki. RCP membatasi izin untuk sumber daya di akun anggota dan dapat memengaruhi izin efektif untuk identitas, termasuk Pengguna root akun AWS, terlepas dari apakah itu milik organisasi Anda. Untuk informasi selengkapnya tentang Organizations dan RCPs, termasuk daftar dukungan Layanan AWS tersebut RCPs, lihat [Kebijakan kontrol sumber daya \(RCPs\)](#) di Panduan AWS Organizations Pengguna.
- **Kebijakan sesi** – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya, lihat [Kebijakan sesi](#) dalam Panduan Pengguna IAM.

## Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat [Logika evaluasi kebijakan](#) di Panduan Pengguna IAM.

## Bagaimana Layanan AWS bekerja dengan IAM

Untuk mendapatkan tampilan tingkat tinggi tentang cara Layanan AWS kerja dengan sebagian besar fitur IAM, lihat [AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM](#).

Untuk mempelajari cara menggunakan yang spesifik Layanan AWS dengan IAM, lihat bagian keamanan dari Panduan Pengguna layanan yang relevan.

## Memecahkan masalah AWS identitas dan akses

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan AWS dan IAM.

### Topik

- [Saya tidak berwenang untuk melakukan tindakan di AWS](#)
- [Saya tidak berwenang untuk melakukan iam: PassRole](#)
- [Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses AWS sumber daya saya](#)

## Saya tidak berwenang untuk melakukan tindakan di AWS

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya `my-example-widget` rekaan, tetapi tidak memiliki izin `aws:GetWidget` rekaan.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
aws:GetWidget on resource: my-example-widget
```

Dalam hal ini, kebijakan untuk pengguna `mateojackson` harus diperbarui untuk mengizinkan akses ke sumber daya `my-example-widget` dengan menggunakan tindakan `aws:GetWidget`.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

## Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan yang tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran AWS.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol tersebut untuk melakukan tindakan di AWS. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

## Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses AWS sumber daya saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACLs), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mempelajari apakah AWS mendukung fitur-fitur ini, lihat [Bagaimana Layanan AWS bekerja dengan IAM](#).

- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat [Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki](#) di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat [Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga](#) dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat [Menyediakan akses ke pengguna terautentikasi eksternal \(federasi identitas\)](#) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM](#).

## Validasi Kepatuhan untuk AWS Produk atau Layanan ini

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat [Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS](#) dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat [Program AWS Kepatuhan Program AWS](#) .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat [Mengunduh Laporan di AWS Artifact](#) .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- [Kepatuhan dan Tata Kelola Keamanan](#) – Panduan implementasi solusi ini membahas pertimbangan arsitektur serta memberikan langkah-langkah untuk menerapkan fitur keamanan dan kepatuhan.
- [Referensi Layanan yang Memenuhi Syarat HIPAA](#) — Daftar layanan yang memenuhi syarat HIPAA. Tidak semua memenuhi Layanan AWS syarat HIPAA.
- [AWS Sumber Daya AWS](#) — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- [AWS Panduan Kepatuhan Pelanggan](#) - Memahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan ini merangkum praktik terbaik untuk mengamankan Layanan AWS dan memetakan panduan untuk kontrol keamanan di berbagai kerangka kerja (termasuk Institut Standar dan Teknologi Nasional (NIST), Dewan Standar Keamanan Industri Kartu Pembayaran (PCI), dan Organisasi Internasional untuk Standardisasi (ISO)).

- [Mengevaluasi Sumber Daya dengan Aturan](#) dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- [AWS Security Hub](#)— Ini Layanan AWS memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS. Security Hub menggunakan kontrol keamanan untuk sumber daya AWS Anda serta untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat [Referensi kontrol Security Hub](#).
- [Amazon GuardDuty](#) — Ini Layanan AWS mendeteksi potensi ancaman terhadap beban kerja Akun AWS, kontainer, dan data Anda dengan memantau lingkungan Anda untuk aktivitas yang mencurigakan dan berbahaya. GuardDuty dapat membantu Anda mengatasi berbagai persyaratan kepatuhan, seperti PCI DSS, dengan memenuhi persyaratan deteksi intrusi yang diamanatkan oleh kerangka kerja kepatuhan tertentu.
- [AWS Audit Manager](#)Ini Layanan AWS membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

AWS Produk atau layanan ini mengikuti [model tanggung jawab bersama](#) melalui layanan Amazon Web Services (AWS) tertentu yang didukungnya. Untuk informasi keamanan AWS layanan, lihat [halaman dokumentasi keamanan AWS layanan](#) dan [AWS layanan yang berada dalam lingkup upaya AWS kepatuhan oleh program kepatuhan](#).

## Ketahanan untuk AWS Produk atau Layanan ini

Infrastruktur AWS global dibangun di sekitar Wilayah AWS dan Availability Zones.

Wilayah AWS menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan.

Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang AWS Wilayah dan Availability Zone, lihat [Infrastruktur AWS Global](#).

AWS Produk atau layanan ini mengikuti [model tanggung jawab bersama](#) melalui layanan Amazon Web Services (AWS) tertentu yang didukungnya. Untuk informasi keamanan AWS layanan, lihat [halaman dokumentasi keamanan AWS layanan](#) dan [AWS layanan yang berada dalam lingkup upaya AWS kepatuhan oleh program kepatuhan](#).

## Keamanan Infrastruktur untuk AWS Produk atau Layanan ini

AWS Produk atau layanan ini menggunakan layanan terkelola, dan karenanya dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat [Keamanan AWS Cloud](#). Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat [Perlindungan Infrastruktur dalam Kerangka Kerja](#) yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses AWS Produk atau Layanan ini melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Sandi cocok dengan sistem kerahasiaan maju sempurna (perfect forward secrecy, PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan prinsipal IAM. Atau Anda dapat menggunakan [AWS Security Token Service](#) (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

AWS Produk atau layanan ini mengikuti [model tanggung jawab bersama](#) melalui layanan Amazon Web Services (AWS) tertentu yang didukungnya. Untuk informasi keamanan AWS layanan, lihat [halaman dokumentasi keamanan AWS layanan](#) dan [AWS layanan yang berada dalam lingkup upaya AWS kepatuhan oleh program kepatuhan](#).

# Riwayat dokumen

Topik ini menjelaskan perubahan penting pada Panduan AWS SDK for Kotlin Pengembang selama sejarahnya.

Perubahan	Deskripsi	Tanggal
<a href="#">Perlindungan integritas data dengan checksum</a>	Konten diperbarui dengan rincian tentang perhitungan checksum otomatis.	Januari 16, 2025
<a href="#">Perbarui konten rantai penyedia kredensyal default</a>	<a href="#">Rantai penyedia kredensyal default.</a>	Januari 15, 2025
<a href="#">Perbarui contoh file build</a>	Tampilkan elemen file build seperti yang dihasilkan oleh Gradle versi 8.11.1. Tampilkan penggunaan BOM. Sematkan tautan ke artefak versi terbaru.	Desember 18, 2024
<a href="#">Tambahkan topik DynamoDB Mapper (Pratinjau Pengembangan)</a>	<a href="#">Memetakan kelas ke item DynamoDB dengan menggunakan DynamoDB Mapper (Pratinjau Pengembangan)</a>	Oktober 29, 2024
<a href="#">Perbarui nama bucket Amazon S3</a>	<a href="#">Checksum Amazon S3 dengan AWS SDK for Kotlin</a>	September 30, 2024
<a href="#">Tambahkan informasi Mesin OkHttp 4</a>	<a href="#">Tentukan tipe mesin HTTP</a>	September 26, 2024
<a href="#">Menambahkan informasi tentang endpoint AWS berbasis akun untuk DynamoDB</a>	<a href="#">Gunakan AWS titik akhir berbasis akun</a>	September 24, 2024



<a href="#">Menambahkan topik Pemecahan Masalah FAQs</a>	<a href="#">Pemecahan Masalah FAQs</a>	September 18, 2024
<a href="#">Perbarui contoh OpenTelemetry konfigurasi dan konfigurasi penyedia telemetri global default</a>	<a href="#">Observabilitas</a>	2 Mei 2024
<a href="#">Berikan detail lebih lanjut tentang proses pembuatan klien layanan</a>	<a href="#">Buat klien layanan</a>	Maret 14, 2024
<a href="#">Tambahkan topik Titik Akses Multi-Wilayah</a>	<a href="#">Bekerja dengan Titik Akses Multi-Wilayah Amazon S3 dengan menggunakan SDK untuk Kotlin</a>	Februari 6, 2024
<a href="#">Tambahkan instruksi katalog versi Gradle</a>	<a href="#">Katalog versi Gradle (tab)</a>	Desember 19, 2023
<a href="#">Rilis Ketersediaan Umum</a>	<a href="#">AWS SDK for Kotlin Panduan Pengembang</a>	27 November 2023
<a href="#">Perbarui bagian konfigurasi titik akhir klien berdasarkan pembaruan SDK</a>	<a href="#">Titik akhir klien</a>	Agustus 25, 2023
<a href="#">Checksum Amazon S3</a>	Bagian ditambahkan tentang cara menggunakan checksum fleksibel dengan Amazon S3.	Agustus 14, 2023
<a href="#">Tambahkan topik Observabilitas</a>	<a href="#">Observabilitas</a>	3 Agustus 2023
<a href="#">Tambahkan topik yang membahas percobaan ulang</a>	<a href="#">Mencoba lagi</a>	7 Juli 2023

<a href="#">Perbarui bagian konfigurasi klien HTTP berdasarkan pembaruan SDK</a>	<a href="#">Konfigurasi klien HTTP</a>	6 Juni 2023
<a href="#">Tambahkan topik presigning HTTP</a>	<a href="#">Permintaan pra-penan datanganan</a>	Juni 2, 2023
<a href="#">Tambahkan topik pencegat HTTP</a>	<a href="#">Pencegat HTTP</a>	22 Mei 2023
<a href="#">Support untuk penyegaran token otomatis</a>	Perbarui <a href="#">instruksi untuk akses masuk tunggal</a> .	18 Mei 2023
<a href="#">Checksum Amazon S3</a>	Tambahkan bagian yang menjelaskan cara <a href="#">menggunakan checksum dengan Amazon S3</a> .	15 Mei 2023
<a href="#">Ganti konfigurasi klien layanan</a>	Tambahkan bagian yang menjelaskan cara <a href="#">mengganti konfigurasi klien layanan dan menjelaskan bagaimana sumber daya terpengaruh</a> .	8 Mei 2023
<a href="#">Menerapkan versi TLS minimum</a>	Tambahkan bagian yang menjelaskan opsi untuk <a href="#">menerapkan versi TLS minimum</a> .	3 Mei 2023
<a href="#">Konfigurasi titik akhir klien</a>	Tambahkan topik yang membahas konfigurasi <a href="#">titik akhir klien</a> .	April 7, 2023
<a href="#">Pembaruan praktik terbaik IAM</a>	Memperbarui panduan untuk menyelaraskan dengan praktik terbaik IAM. Untuk informasi lebih lanjut, lihat <a href="#">Praktik terbaik keamanan di IAM</a> .	22 Maret 2023

<a href="#">Tambahkan contoh file proyek Maven</a>	Tampilkan contoh file proyek Maven selain file proyek di Gradle dalam topik <a href="#">Set up</a> .	Desember 2, 2022
<a href="#">Merevisi isi Pratinjau Panduan Pengembang</a>	Konten diperbarui untuk mencerminkan pekerjaan pengembangan terbaru	4 Oktober 2022
<a href="#">AWS SDK for Kotlin Rilis Pratinjau Pengembang</a>	<a href="#">AWS SDK for Kotlin</a>	2 Desember 2021
<a href="#">AWS SDK for Kotlin pelepasan alfa</a>	<a href="#">Mengumumkan rilis AWS SDK for Kotlin alpha baru</a>	Agustus 30, 2021

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.