

Panduan Pengembang untuk versi 2.x

# AWS SDK for Java 2.x



# AWS SDK for Java 2.x: Panduan Pengembang untuk versi 2.x

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 dapat menyebabkan kebingungan di antara para pelanggan, atau dengan cara apa pun yang menghina atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan kekayaan masing-masing pemiliknya, yang mungkin berafiliasi, terkait dengan, atau disponsori oleh Amazon, atau mungkin tidak.

---

# Table of Contents

Panduan Pengembang - AWS SDK for Java 2.x .....	1
Memulai dengan SDK .....	1
Kembangkan aplikasi seluler .....	1
Pemeliharaan dan dukungan untuk versi utama SDK .....	1
Sumber daya tambahan .....	2
Berkontribusi pada SDK .....	2
Mulai tutorial .....	3
Langkah 1: Siapkan untuk tutorial ini .....	3
Langkah 2: Buat proyek .....	3
Langkah 3: Tulis kodenya .....	8
Langkah 4: Bangun dan jalankan aplikasi .....	12
Berhasil .....	13
Pembersihan .....	13
Langkah berikutnya .....	13
Pengaturan .....	15
Gambaran umum pengaturan .....	15
Kemampuan masuk ke portal AWS akses .....	16
Menyiapkan akses masuk tunggal untuk SDK .....	16
Masuk menggunakan AWS CLI .....	17
Instal Java dan alat build .....	18
Opsi otentikasi tambahan .....	18
Siapkan proyek Apache Maven .....	18
Prasyarat .....	18
Buat proyek Maven .....	19
Konfigurasikan compiler Java untuk Maven .....	20
Mendeklarasikan SDK sebagai dependensi .....	21
Tetapkan dependensi untuk modul SDK .....	22
Bangun proyek Anda .....	24
Siapkan proyek Gradle .....	25
Menyiapkan proyek GraAlvm Native Image .....	31
Prasyarat .....	31
Buat proyek menggunakan arketipe .....	32
Membangun gambar asli .....	33
Gunakan SDK .....	34

Bekerja dengan klien layanan .....	34
Buat klien layanan .....	34
Konfigurasi klien default .....	35
Konfigurasikan klien layanan .....	35
Membuat permintaan .....	36
Menangani tanggapan .....	36
Tutup klien layanan .....	37
Menangani pengecualian .....	37
Gunakan pelayan .....	38
Klien HTTP .....	39
Percobaan ulang .....	39
Timeout .....	39
Pencegat eksekusi .....	40
Informasi tambahan .....	40
Memberikan kredensi sementara ke SDK .....	40
Konfigurasikan akses ke kredensial sementara .....	41
Rantai penyedia kredensi default .....	42
Menggunakan penyedia kredensial tertentu atau rantai penyedia .....	44
Gunakan profil .....	45
Memuat kredensi sementara dari proses eksternal .....	48
Menyediakan kredensi sementara dalam kode .....	51
Baca kredensi peran IAM di Amazon EC2 .....	55
Gunakan Wilayah AWS .....	56
Konfigurasikan secara eksplisit Wilayah AWS .....	57
Tentukan Wilayah dari lingkungan .....	57
Periksa ketersediaan layanan .....	59
Pilih titik akhir tertentu .....	59
Kurangi waktu startup SDK untuk AWS Lambda .....	59
Gunakan SDK <code>URLConnectionHttpClient</code> .....	60
Gunakan SDK <code>AwsCrtAsyncHttpClient</code> .....	60
Hapus dependensi klien HTTP yang tidak digunakan .....	61
Konfigurasikan klien layanan untuk memintas pencarian .....	62
Inisialisasi klien SDK di luar penanganan fungsi Lambda .....	63
Minimalkan injeksi ketergantungan .....	64
Gunakan penargetan Maven Archetype AWS Lambda .....	64
Lambda SnapStart .....	64

Perubahan versi 2.x yang memengaruhi waktu startup .....	64
Sumber daya tambahan .....	65
Klien HTTP .....	65
Klien yang tersedia .....	65
Rekomendasi klien .....	66
Default cerdas .....	69
Dukungan proxy .....	71
Konfigurasi klien HTTP berbasis Apache .....	73
Konfigurasi klien HTTP berbasis URLConnection .....	78
Konfigurasi klien HTTP berbasis Netty .....	84
Konfigurasi AWS klien HTTP berbasis CRT .....	89
Penanganan pengecualian .....	100
Mengapa pengecualian yang tidak dicentang? .....	100
AwsServiceException (dan subclass) .....	100
SdkClientException .....	101
Pengecualian dan perilaku coba lagi .....	102
Pencatatan log .....	102
File konfigurasi Log4j 2 .....	102
Tambahkan ketergantungan logging .....	103
Kesalahan dan peringatan khusus SDK .....	103
Pencatatan ringkasan permintaan/tanggapan .....	104
Penebangan kawat verbose .....	105
Mengatur JVM TTL untuk pencarian nama DNS .....	110
Cara mengatur JVM TTL .....	111
Praktik terbaik .....	111
Menggunakan kembali klien SDK .....	112
Tutup aliran masukan .....	112
Tune konfigurasi HTTP .....	112
Gunakan OpenSSL untuk Netty .....	113
Konfigurasi batas waktu API .....	113
Gunakan metrik .....	114
Gunakan fitur SDK .....	115
Fitur umum .....	115
Fitur khusus layanan .....	115
Bekerja dengan hasil paginasi .....	115
pagination sinkron .....	116

pagination asinkron .....	119
Polling untuk negara sumber daya .....	124
Prasyarat .....	125
Menggunakan pelayan .....	125
Konfigurasi pelayan .....	126
Contoh kode .....	127
Gunakan pemrograman asinkron .....	127
Operasi non-streaming .....	128
Operasi streaming .....	131
Operasi lanjutan .....	134
Bekerja dengan HTTP/2 .....	136
Gunakan metrik SDK .....	136
Prasyarat .....	136
Cara mengaktifkan pengumpulan metrik .....	137
Kapan metrik tersedia? .....	138
Informasi apa yang dikumpulkan? .....	139
Bagaimana saya bisa menggunakan informasi ini? .....	139
Metrik klien layanan .....	139
Bekerja dengan Layanan AWS .....	145
CloudWatch .....	145
Dapatkan metrik dari CloudWatch .....	146
Publikasikan data metrik kustom ke CloudWatch .....	148
Bekerja dengan CloudWatch alarm .....	150
Gunakan CloudWatch Acara Amazon .....	154
AWSlayanan basis data .....	158
Amazon DynamoDB .....	158
Amazon RDS .....	159
Amazon Redshift .....	159
Amazon Aurora Tanpa Server v1 .....	159
Amazon DocumentDB .....	160
DynamoDB .....	160
Bekerja dengan tabel di DynamoDB .....	160
Bekerja dengan item di DynamoDB .....	170
Peta objek ke item DynamoDB .....	178
Amazon EC2 .....	296
Kelola Amazon EC2 instance .....	296

Zona Penggunaan Wilayah AWS dan Ketersediaan .....	303
Bekerja dengan kelompok keamanan di Amazon EC2 .....	307
Bekerja dengan metadata instans Amazon EC2 .....	311
IAM .....	318
Kelola kunci IAM akses .....	318
Kelola IAM Pengguna .....	324
Buat kebijakan IAM .....	329
Bekerja dengan IAM kebijakan .....	336
Bekerja dengan sertifikat IAM server .....	343
Kinesis .....	347
Berlangganan Amazon Kinesis Data Streams .....	348
Lambda .....	359
Memanggil fungsi Lambda .....	359
Daftar fungsi Lambda .....	360
Hapus fungsi Lambda .....	361
Amazon S3 .....	362
Gunakan titik akses atau Titik Akses Multi-Wilayah .....	363
Operasi bucket .....	364
Operasi objek .....	371
URL yang telah ditandatangani sebelumnya .....	381
Akses Lintas Wilayah .....	390
Checksum .....	391
Gunakan klien S3 yang berkinerja .....	392
Transfer file dan direktori .....	395
Amazon SNS .....	403
Buat topik .....	403
Buat daftar Amazon SNS topik Anda .....	404
Berlangganan titik akhir ke suatu topik .....	405
Publikasikan pesan ke suatu topik .....	406
Berhenti berlangganan titik akhir dari suatu topik .....	407
Hapus topik .....	408
Amazon SQS .....	409
Operasi antrian .....	409
Operasi pesan .....	413
Amazon Transcribe .....	416
Siapkan mikrofon .....	416

Buat penerbit .....	417
Buat klien dan mulai streaming .....	420
Informasi lain .....	416
Contoh kode .....	422
Tindakan dan skenario .....	422
API Gateway .....	424
Application Auto Scaling .....	428
Pengontrol Pemulihan Aplikasi .....	437
Aurora .....	439
Auto Scaling .....	474
Amazon Bedrock .....	536
Waktu Jalan Amazon Bedrock .....	542
CloudFront .....	568
CloudWatch .....	587
CloudWatch Acara .....	638
CloudWatch Log .....	644
Identitas Amazon Cognito .....	654
Penyedia Identitas Amazon Cognito .....	662
Amazon Comprehend .....	689
DynamoDB .....	700
Amazon EC2 .....	766
Amazon ECS .....	837
Penyeimbang Beban Elastis .....	851
MediaStore .....	896
OpenSearch Layanan .....	911
EventBridge .....	920
Forecast .....	951
AWS Glue .....	965
HealthImaging .....	988
IAM .....	1014
AWS IoT .....	1099
AWS IoT data .....	1126
Amazon Keyspaces .....	1129
Kinesis .....	1155
AWS KMS .....	1168
Lambda .....	1186



MediaConvert .....	1210
Migration Hub .....	1233
Amazon Personalize .....	1246
Kejadian Amazon Personalize .....	1275
Waktu Aktif Amazon Personalize .....	1279
Amazon Pinpoint .....	1283
API SMS dan Suara Amazon Pinpoint .....	1328
Amazon Polly .....	1331
Amazon RDS .....	1337
Amazon Redshift .....	1378
Amazon Rekognition .....	1383
Pendaftaran domain Route 53 .....	1451
Amazon S3 .....	1474
S3 Glacier .....	1581
SageMaker .....	1597
Secrets Manager .....	1626
Amazon SES .....	1639
Amazon SES API v2 .....	1651
Amazon SNS .....	1655
Amazon SQS .....	1703
Step Functions .....	1723
AWS STS .....	1747
AWS Support .....	1750
Systems Manager .....	1773
Amazon Textract .....	1783
Amazon Transcribe .....	1793
Contoh lintas layanan .....	1810
Membangun aplikasi untuk mengirimkan data ke tabel DynamoDB .....	1810
Membangun chatbot Amazon Lex .....	1811
Membangun aplikasi Amazon SNS .....	1811
Buat aplikasi perpesanan .....	1812
Membuat aplikasi nirserver untuk mengelola foto .....	1812
Membuat aplikasi web untuk melacak data DynamoDB .....	1813
Buat aplikasi web untuk melacak data Amazon Redshift .....	1813
Buat pelacak butir kerja Aurora Serverless .....	1813
Buat aplikasi untuk menganalisis umpan balik pelanggan .....	1814

Mendeteksi APD dalam gambar .....	1815
Mendeteksi objek dalam gambar .....	1815
Mendeteksi orang dan objek dalam video .....	1816
Publikasikan pesan ke antrian .....	1816
Menggunakan API Gateway untuk menginvokasi fungsi Lambda .....	1817
Menggunakan Step Functions untuk menginvokasi fungsi Lambda .....	1817
Menggunakan peristiwa terjadwal untuk menginvokasi fungsi Lambda .....	1818
Keamanan .....	1819
Perlindungan data .....	1819
Keamanan Lapisan Pengangkutan (TLS) .....	1821
Periksa versi TLS .....	1821
Menegakkan versi TLS .....	1822
Migrasi ke TLS 1.2 .....	1822
Manajemen Identitas dan Akses .....	1822
Audiens .....	1823
Mengautentikasi dengan identitas .....	1823
Mengelola akses menggunakan kebijakan .....	1827
Bagaimana Layanan AWS bekerja dengan IAM .....	1830
Memecahkan masalah AWS identitas dan akses .....	1830
Validasi Kepatuhan .....	1832
Ketangguhan .....	1833
Keamanan Infrastruktur .....	1834
Migrasi ke versi 2 .....	1835
Apa yang baru di versi 2 .....	1835
tep-by-step Instruksi S .....	1836
Ikhtisar langkah-langkah .....	1836
Contoh migrasi .....	1838
Apa yang berbeda antara 1.x dan 2.x .....	1848
Perubahan nama Package .....	1849
Menambahkan versi 2.x ke proyek Anda .....	1849
PojOS yang tidak dapat diubah .....	1850
Metode setter dan getter .....	1851
Nama kelas model .....	1851
Perpustakaan dan utilitas .....	1852
Perubahan klien .....	1853
Perubahan penyedia kredensial .....	1898

---

Perubahan wilayah .....	1906
Operasi, permintaan, dan tanggapan berubah .....	1908
Perubahan pengecualian .....	1909
Perubahan serialisasi .....	1911
Perubahan khusus layanan .....	1912
Perubahan file profil .....	1917
Konfigurasi eksternal .....	1919
Pelayan .....	1922
Manajer Transfer S3 .....	1925
Utilitas metadata EC2 .....	1932
CloudFront prepenandatanganan .....	1939
Penguraian URI S3 .....	1943
Gunakan SDK for Java 1.x and 2.x side-by-side .....	1945
Kunci OpenPGP .....	1947
Kunci saat ini .....	1947
Riwayat dokumen .....	1949
.....	mcmlvi

# Panduan Pengembang - AWS SDK for Java 2.x

AWS SDK for Java ini menyediakan Java API untuk Layanan AWS. Menggunakan SDK, Anda dapat membangun aplikasi Java yang bekerja dengan Amazon S3, Amazon EC2, DynamoDB, dan banyak lagi.

AWS SDK for Java 2.x adalah penulisan ulang utama dari basis kode versi 1.x. Ini dibangun di atas Java 8+ dan menambahkan beberapa fitur yang sering diminta. Ini termasuk dukungan untuk I/O non-pemblokiran dan kemampuan untuk menyambungkan implementasi HTTP yang berbeda saat runtime.

Kami secara teratur menambahkan dukungan untuk layanan baru ke AWS SDK for Java. Untuk daftar perubahan dan fitur dalam versi tertentu, lihat [log perubahan](#).

## Memulai dengan SDK

Jika Anda siap untuk langsung menggunakan SDK, ikuti tutorialnya. [Mulai tutorial](#)

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

Jika saat ini Anda menggunakan versi 1.x SDK for Java, lihat [Migrasi ke versi 2 untuk panduan khusus](#).

Untuk informasi tentang membuat permintaan ke Amazon S3, DynamoDB, Amazon EC2 dan lainnya Layanan AWS, lihat [Menggunakan SDK for Java](#) dan [Bekerja dengan Layanan AWS](#).

## Kembangkan aplikasi seluler

Jika Anda seorang pengembang aplikasi seluler, Amazon Web Services sediakan [AWS Amplify](#) kerangka kerja.

## Pemeliharaan dan dukungan untuk versi utama SDK

Untuk informasi tentang pemeliharaan dan dukungan untuk versi utama SDK dan dependensi yang mendasarinya, lihat topik berikut di Panduan Referensi [AWS SDK dan Alat](#):

- [AWS Kebijakan Pemeliharaan SDK dan Alat](#)

- [AWSMatriks Dukungan Versi SDK dan Alat](#)

## Sumber daya tambahan

Selain panduan ini, berikut ini adalah sumber daya online yang berharga bagi AWS SDK for Java pengembang:

- [AWS SDK for Java2.x API Referensi](#)
- [Blog pengembang Java](#)
- [Topik pengembangan Java di AWS re:Post](#)
- [Sumber SDK](#) pada GitHub
- [AWSPustaka Contoh Kode SDK](#)
- [@awsforjava \(Twitter\)](#)

## Berkontribusi pada SDK

Pengembang juga dapat menyumbangkan umpan balik melalui saluran berikut:

- Kirim masalah pada GitHub:
  - [Kirim masalah dokumentasi Panduan Pengembang](#)
  - [Kirim masalah SDK](#)
- [Bergabunglah dengan obrolan informal tentang SDK di saluran AWS SDK for Java gitter 2.x](#)

# Memulai dengan AWS SDK for Java 2.x

AWS SDK for Java 2.x ini menyediakan Java API untuk Amazon Web Services (AWS). Menggunakan SDK, Anda dapat membangun aplikasi Java yang bekerja dengan Amazon S3, Amazon EC2, DynamoDB, dan banyak lagi.

Tutorial ini menunjukkan cara menggunakan [Apache Maven](#) untuk menentukan dependensi untuk SDK for Java 2.x dan kemudian menulis kode yang terhubung ke untuk mengunggah file. Amazon S3

Ikuti langkah-langkah berikut untuk menyelesaikan tutorial ini:

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

## Langkah 1: Siapkan untuk tutorial ini

Sebelum Anda memulai tutorial ini, Anda memerlukan yang berikut:

- Izin untuk mengakses Amazon S3
- Lingkungan pengembangan Java yang dikonfigurasi untuk mengakses Layanan AWS menggunakan sistem masuk tunggal ke AWS IAM Identity Center

Gunakan instruksi [???](#) untuk mengatur tutorial ini. Setelah Anda [mengkonfigurasi lingkungan pengembangan Anda dengan akses masuk tunggal](#) untuk Java SDK dan Anda memiliki [sesi portal AWS akses aktif](#), lanjutkan dengan Langkah 2 dari tutorial ini.

## Langkah 2: Buat proyek

Untuk membuat proyek untuk tutorial ini, Anda menjalankan perintah Maven yang meminta Anda untuk masukan tentang cara mengkonfigurasi proyek. Setelah semua input dimasukkan dan dikonfirmasi, Maven selesai membangun proyek dengan membuat `pom.xml` dan membuat file Java rintisan.

1. Buka terminal atau jendela prompt perintah dan arahkan ke direktori pilihan Anda, misalnya, Home folder Anda Desktop atau.
2. Masukkan perintah berikut di terminal dan tekan `Enter`.

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

3. Masukkan nilai yang tercantum di kolom kedua untuk setiap prompt.

Prompt	Nilai untuk masuk
Define value for property 'service':	s3
Define value for property 'httpClient' :	apache-client
Define value for property 'nativeImage' :	false
Define value for property 'credentialProvider'	identity-center
Define value for property 'groupId':	org.example
Define value for property 'artifactId':	getstarted
Define value for property 'version' 1.0-SNAPSHOT:	<Enter>
Define value for property 'package' org.example:	<Enter>

4. Setelah nilai terakhir dimasukkan, Maven mencantumkan pilihan yang Anda buat. Konfirmasikan dengan memasukkan `Y` atau memasukkan kembali nilai dengan memasukkan `N`.

Maven membuat folder proyek bernama `getstarted` berdasarkan `artifactId` nilai yang Anda masukkan. Di dalam `getstarted` folder, temukan `README.md` file yang dapat Anda tinjau, `pom.xml` file, dan `src` direktori.

Maven membangun pohon direktori berikut.

```
getstarted
### README.md
### pom.xml
### src
  ### main
  #   ### java
  #   #   ### org
  #   #   ### example
  #   #   ### App.java
  #   #   ### DependencyFactory.java
  #   #   ### Handler.java
  #   ### resources
  #   ### simplelogger.properties
  ### test
  #   ### java
  #   #   ### org
  #   #   ### example
  #   #   ### HandlerTest.java

10 directories, 7 files
```

Berikut ini menunjukkan isi dari file `pom.xml` proyek.

## **pom.xml**

`dependencyManagement` bagian ini berisi ketergantungan ke AWS SDK for Java 2.x dan `dependencies` bagian tersebut memiliki ketergantungan untuk Amazon S3. Proyek ini menggunakan Java 1.8 karena 1.8 nilai dalam `maven.compiler.source` dan `maven.compiler.target` properti.

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



```

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

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

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

```

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>sso</artifactId> <----- Required for identity center
authentication.
</dependency>

<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>ssooidc</artifactId> <----- Required for identity center
authentication.
</dependency>

<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>apache-client</artifactId> <----- HTTP client specified.
  <exclusions>
    <exclusion>
      <groupId>commons-logging</groupId>
      <artifactId>commons-logging</artifactId>
    </exclusion>
  </exclusions>
</dependency>

<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>${slf4j.version}</version>
</dependency>

<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-simple</artifactId>
  <version>${slf4j.version}</version>
</dependency>

<!-- Needed to adapt Apache Commons Logging used by Apache HTTP Client to Slf4j
to avoid
ClassNotFoundException: org.apache.commons.logging.impl.LogFactoryImpl during
runtime -->
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>jcl-over-slf4j</artifactId>
  <version>${slf4j.version}</version>
```

```
    </dependency>

    <!-- Test Dependencies -->
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter</artifactId>
      <version>${junit5.version}</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>${maven.compiler.plugin.version}</version>
      </plugin>
    </plugins>
  </build>

</project>
```

## Langkah 3: Tulis kodenya

Kode berikut menunjukkan App kelas yang dibuat oleh Maven. `mainMetode` ini adalah titik masuk ke dalam aplikasi, yang menciptakan sebuah instance dari `Handler` kelas dan kemudian memanggil `sendRequest` metodenya.

### App kelas

```
package org.example;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class App {
    private static final Logger logger = LoggerFactory.getLogger(App.class);

    public static void main(String... args) {
        logger.info("Application starts");

        Handler handler = new Handler();
```

```
        handler.sendRequest();

        logger.info("Application ends");
    }
}
```

DependencyFactoryKelas yang dibuat oleh Maven berisi metode `s3Client` pabrik yang membangun dan mengembalikan sebuah [S3Client](#) instance. `S3Client` instance menggunakan instance dari klien HTTP berbasis Apache. Ini karena Anda menentukan `apache-client` kapan Maven meminta Anda untuk klien HTTP mana yang akan digunakan.

DependencyFactoryHal ini ditunjukkan dalam kode berikut.

## DependencyFactory kelas

```
package org.example;

import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;

/**
 * The module containing all dependencies required by the {@link Handler}.
 */
public class DependencyFactory {

    private DependencyFactory() {}

    /**
     * @return an instance of S3Client
     */
    public static S3Client s3Client() {
        return S3Client.builder()
            .httpClientBuilder(ApacheHttpClient.builder())
            .build();
    }
}
```

HandlerKelas berisi logika utama program Anda. Ketika sebuah instance Handler dibuat di App kelas, DependencyFactory melengkapi klien S3Client layanan. Kode Anda menggunakan S3Client instance untuk memanggil layanan Amazon S3.

Maven menghasilkan `Handler` kelas berikut dengan komentar. *TODO* Langkah selanjutnya dalam tutorial menggantikan kode *TODO* dengan.

## **Handler** kelas, yang dihasilkan oleh Maven

```
package org.example;

import software.amazon.awssdk.services.s3.S3Client;

public class Handler {
    private final S3Client s3Client;

    public Handler() {
        s3Client = DependencyFactory.s3Client();
    }

    public void sendRequest() {
        // TODO: invoking the api calls using s3Client.
    }
}
```

Untuk mengisi logika, ganti seluruh isi `Handler` kelas dengan kode berikut. `sendRequest` metode diisi dan impor yang diperlukan ditambahkan.

## **Handler** kelas, diimplementasikan

Kode pertama membuat bucket S3 baru dengan bagian terakhir dari nama yang dihasilkan `System.currentTimeMillis()` untuk membuat nama bucket unik.

Setelah membuat bucket dalam `createBucket()` metode, program mengunggah objek menggunakan [putObject](#) metode. `S3Client` Isi objek adalah string sederhana yang dibuat dengan `RequestBody.fromString` metode ini.

Akhirnya, program menghapus objek diikuti oleh bucket dalam `cleanup` metode.

```
package org.example;

import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
```

```
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class Handler {
    private final S3Client s3Client;

    public Handler() {
        s3Client = DependencyFactory.s3Client();
    }

    public void sendRequest() {
        String bucket = "bucket" + System.currentTimeMillis();
        String key = "key";

        createBucket(s3Client, bucket);

        System.out.println("Uploading object...");

        s3Client.putObject(PutObjectRequest.builder().bucket(bucket).key(key)
            .build(),
            RequestBody.fromString("Testing with the {sdk-java}"));

        System.out.println("Upload complete");
        System.out.printf("%n");

        cleanUp(s3Client, bucket, key);

        System.out.println("Closing the connection to {S3}");
        s3Client.close();
        System.out.println("Connection closed");
        System.out.println("Exiting...");
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            s3Client.createBucket(CreateBucketRequest
                .builder()
                .bucket(bucketName)
                .build());
            System.out.println("Creating bucket: " + bucketName);
            s3Client.waiter().waitUntilBucketExists(HeadBucketRequest.builder()
```

```

        .bucket(bucketName)
        .build());
    System.out.println(bucketName + " is ready.");
    System.out.printf("%n");
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void cleanUp(S3Client s3Client, String bucketName, String keyName) {
    System.out.println("Cleaning up...");
    try {
        System.out.println("Deleting object: " + keyName);
        DeleteObjectRequest deleteObjectRequest =
DeleteObjectRequest.builder().bucket(bucketName).key(keyName).build();
s3Client.deleteObject(deleteObjectRequest);
        System.out.println(keyName + " has been deleted.");
        System.out.println("Deleting bucket: " + bucketName);
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder().bucket(bucketName).build();
s3Client.deleteBucket(deleteBucketRequest);
        System.out.println(bucketName + " has been deleted.");
        System.out.printf("%n");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Cleanup complete");
    System.out.printf("%n");
}
}

```

## Langkah 4: Bangun dan jalankan aplikasi

Setelah proyek dibuat dan berisi Handler kelas lengkap, bangun dan jalankan aplikasi.

1. Pastikan Anda memiliki sesi IAM Identity Center yang aktif. Untuk melakukannya, jalankan AWS Command Line Interface perintah `aws sts get-caller-identity` dan periksa responsnya. Jika Anda tidak memiliki sesi aktif, lihat [bagian ini](#) untuk petunjuk.
2. Buka terminal atau jendela prompt perintah dan arahkan ke direktori proyek `Andagetstarted`.

### 3. Gunakan perintah berikut untuk membangun proyek Anda:

```
mvn clean package
```

### 4. Gunakan perintah berikut untuk menjalankan aplikasi.

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

Untuk melihat bucket dan objek baru yang dibuat program, lakukan langkah-langkah berikut.

1. `MasukHandler.java`, komentari baris `cleanUp(s3Client, bucket, key)` dalam `sendRequest` metode dan simpan file.
2. Membangun kembali proyek dengan menjalankan `mvn clean package`.
3. Jalankan kembali `mvn exec:java -Dexec.mainClass="org.example.App"` untuk mengunggah objek teks sekali lagi.
4. Masuk ke [konsol S3](#) untuk melihat objek baru di bucket yang baru dibuat.

Setelah Anda melihat file, hapus objek, dan kemudian hapus ember.

## Berhasil

Jika proyek Maven Anda dibangun dan berjalan tanpa kesalahan, maka selamat! Anda telah berhasil membangun aplikasi Java pertama Anda menggunakan SDK for Java 2.x.

## Pembersihan

Untuk membersihkan sumber daya yang Anda buat selama tutorial ini, lakukan hal berikut:

- Jika Anda belum melakukannya, di [konsol S3](#), hapus objek apa pun dan ember apa pun yang dibuat saat Anda menjalankan aplikasi.
- Hapus folder proyek (`getstarted`).

## Langkah berikutnya

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

- [Bekerja dengan Amazon S3](#)



- [Bekerja dengan yang lain Amazon Web Services](#), seperti [DynamoDB](#), [Amazon EC2](#), dan [berbagai layanan database](#)
- [Gunakan SDK](#)
- [Keamanan untuk AWS SDK for Java](#)

# Siapkan AWS SDK for Java 2.x

Bagian ini memberikan informasi tentang cara mengatur lingkungan pengembangan dan proyek Anda untuk menggunakan AWS SDK for Java 2.x.

## Gambaran umum pengaturan

Untuk berhasil mengembangkan aplikasi yang mengakses Layanan AWS menggunakan AWS SDK for Java, kondisi berikut diperlukan:

- 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](#), [Eclipse](#), atau [NetBeans](#)

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

- 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 bagaimana Anda bisa mendapatkan kredensi sementara untuk SDK for Java 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 dalam topik [otentikasi Pusat Identitas IAM](#) di AWS SDK dan Panduan Referensi 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 for Java 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 AWS SDK dan Panduan Referensi Alat.

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

## Masuk menggunakan AWS CLI

Sebelum menjalankan aplikasi yang mengakses Layanan AWS, Anda memerlukan sesi portal AWS akses aktif agar SDK dapat 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 menjalankannya `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 for Java.

## Instal Java dan alat build

Lingkungan pengembangan Anda membutuhkan yang berikut:

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

## Opsi otentikasi tambahan

Untuk opsi lainnya tentang otentikasi SDK, seperti penggunaan profil dan variabel lingkungan, lihat bagian [konfigurasi](#) di Panduan Referensi AWS SDK dan Alat.

## Siapkan proyek Apache Maven

Anda dapat menggunakan [Apache Maven](#) untuk menyiapkan dan membangun AWS SDK for Java proyek, atau untuk [membangun SDK itu sendiri](#).

## Prasyarat

Untuk menggunakan AWS SDK for Java dengan Maven, Anda memerlukan yang berikut:

- Java 8.0 atau yang lebih baru. Anda dapat mengunduh perangkat lunak Java SE Development Kit terbaru dari <http://www.oracle.com/technetwork/java/javase/downloads/>. Ini AWS SDK for Java juga bekerja dengan [OpenJDK](#) Amazon Corretto dan, distribusi Open Java Development Kit (OpenJDK). [Unduh versi OpenJDK terbaru dari https://openjdk.java.net/install/index.html](https://openjdk.java.net/install/index.html). Unduh versi Amazon Corretto 8 atau Amazon Corretto 11 terbaru [dari Corretto halaman](#).
- Apache Maven. Jika Anda perlu menginstal Maven, buka <http://maven.apache.org/> untuk mengunduh dan menginstalnya.

## Buat proyek Maven

Untuk membuat proyek Maven dari baris perintah, jalankan perintah berikut dari terminal atau jendela prompt perintah.

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

### Note

Ganti `com.example.myapp` dengan namespace paket lengkap aplikasi Anda. Juga ganti `myapp` dengan nama proyek Anda. Ini menjadi nama direktori untuk proyek Anda.

[Untuk menggunakan arketipe versi terbaru, ganti 2.X.X dengan yang terbaru dari pusat Maven.](#)

Perintah ini membuat proyek Maven menggunakan toolkit templating arketipe. Pola dasar menghasilkan perancah untuk proyek penanganan fungsi. AWS Lambda Arketipe proyek ini telah dikonfigurasi sebelumnya untuk dikompilasi dengan Java SE 8 dan menyertakan ketergantungan ke versi SDK for Java 2.x yang ditentukan dengan `-DarchetypeVersion`

Untuk informasi selengkapnya tentang membuat dan mengonfigurasi proyek Maven, lihat Panduan Memulai [Maven](#).

## Konfigurasi compiler Java untuk Maven

Jika Anda membuat proyek menggunakan pola dasar AWS Lambda proyek seperti yang dijelaskan sebelumnya, konfigurasi kompilasi Java sudah dilakukan untuk Anda.

Untuk memverifikasi bahwa konfigurasi ini ada, mulailah dengan membuka `pom.xml` file dari folder proyek yang Anda buat (misalnya, `myapp`) ketika Anda menjalankan perintah sebelumnya. Lihat pada baris 11 dan 12 untuk melihat pengaturan versi kompilasi Java untuk proyek Maven ini, dan pernyataan yang diperlukan dari plugin kompilasi Maven pada baris 71-75.

```
<project>
  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>${maven.compiler.plugin.version}</version>
      </plugin>
    </plugins>
  </build>
</project>
```

Jika Anda membuat proyek dengan pola dasar yang berbeda atau dengan menggunakan metode lain, Anda harus memastikan bahwa plugin kompilasi Maven adalah bagian dari build dan properti sumber dan targetnya disetel ke 1.8 dalam file `pom.xml`

Lihat cuplikan sebelumnya untuk satu cara mengonfigurasi pengaturan yang diperlukan ini.

Atau, Anda dapat mengkonfigurasi konfigurasi kompilasi sebaris dengan deklarasi plugin, sebagai berikut.

```
<project>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
```

```
<configuration>
  <source>1.8</source>
  <target>1.8</target>
</configuration>
</plugin>
</plugins>
</build>
</project>
```

## Mendeklarasikan SDK sebagai dependensi

Untuk menggunakan AWS SDK for Java dalam proyek Anda, Anda perlu mendeklarasikannya sebagai dependensi dalam file proyek Anda. `pom.xml`

Jika Anda membuat proyek menggunakan pola dasar proyek seperti yang dijelaskan sebelumnya, versi terbaru SDK sudah dikonfigurasi sebagai dependensi dalam proyek Anda.

Pola dasar menghasilkan ketergantungan artefak BOM (bill of material) untuk id grup. `software.amazon.awssdk` Dengan BOM, Anda tidak perlu menentukan versi maven untuk dependensi artefak individual yang berbagi id grup yang sama.

Jika Anda membuat proyek Maven dengan cara yang berbeda, konfigurasi versi terbaru SDK untuk proyek Anda dengan memastikan bahwa `pom.xml` file tersebut berisi yang berikut ini.

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



**Note**

Ganti **2.X.X** dalam `pom.xml` file dengan [versi terbaru](#) dari file. AWS SDK for Java 2.x

## Tetapkan dependensi untuk modul SDK

Sekarang setelah Anda mengonfigurasi SDK, Anda dapat menambahkan dependensi untuk satu atau beberapa AWS SDK for Java modul yang akan digunakan dalam proyek Anda.

Meskipun Anda dapat menentukan nomor versi untuk setiap komponen, Anda tidak perlu melakukannya karena Anda sudah mendeklarasikan versi SDK di `dependencyManagement` bagian menggunakan artefak tagihan bahan. Untuk memuat versi berbeda dari modul tertentu, tentukan nomor versi untuk ketergantungannya.

Jika Anda membuat proyek menggunakan pola dasar proyek seperti yang dijelaskan sebelumnya, proyek Anda sudah dikonfigurasi dengan beberapa dependensi. Ini termasuk dependensi untuk penangan AWS Lambda fungsi dan Amazon S3, sebagai berikut.

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

    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>url-connection-client</artifactId>
    </dependency>
  </dependencies>
</project>
```

```

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

```

### Note

Pada pom.xml contoh di atas, dependensi berasal dari s yang berbedagroupId. s3Ketergantungan berasal dari software.amazon.awssdk, sedangkan aws-lambda-java-core dependensi berasal dari com.amazonaws Konfigurasi manajemen ketergantungan BOM memengaruhi artefak untuk software.amazon.awssdk, sehingga diperlukan versi untuk artefak. aws-lambda-java-core Untuk pengembangan penanganan fungsi Lambda menggunakan SDK for Java 2.xaws-lambda-java-core, adalah dependensi yang benar. Namun, jika aplikasi Anda perlu mengelola sumber daya Lambda, menggunakan operasi seperti listFunctions,, dan deleteFunction invokeFunctioncreateFunction, aplikasi Anda memerlukan dependensi berikut.

```

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

```

### Note

s3Ketergantungan mengecualikan dependensi netty-nio-client dan apache-client transitif. Sebagai pengganti salah satu klien HTTP tersebut, arketipe mencakup url-connection-client ketergantungan, yang membantu [mengurangi latensi startup untuk fungsi](#). AWS Lambda

Tambahkan modul ke proyek Anda untuk Layanan AWS dan fitur yang Anda butuhkan untuk proyek Anda. Modul (dependensi) yang dikelola oleh AWS SDK for Java BOM terdaftar di repositori pusat [Maven](#).

**Note**

Anda dapat melihat pom.xml file dari contoh kode untuk menentukan dependensi mana yang Anda butuhkan untuk proyek Anda. [Misalnya, jika Anda tertarik dengan dependensi untuk layanan DynamoDB, lihat contoh ini dari Repositori Contoh Kode di AWS GitHub](#) (Cari pom.xml file di bawah [/javav2/example\\_code/dynamodb.](#))

## Membangun seluruh SDK ke dalam proyek Anda

Untuk mengoptimalkan aplikasi Anda, kami sangat menyarankan agar Anda hanya menarik komponen yang Anda butuhkan, bukan seluruh SDK. Namun, untuk membangun keseluruhan AWS SDK for Java ke dalam proyek Anda, nyatakan dalam pom.xml file Anda, sebagai berikut.

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

## Bangun proyek Anda

Setelah Anda mengkonfigurasi pom.xml file, Anda dapat menggunakan Maven untuk membangun proyek Anda.

Untuk membangun proyek Maven Anda dari baris perintah, buka jendela terminal atau command prompt, arahkan ke direktori proyek Anda (misalnya,myapp), masukkan atau tempel perintah berikut, lalu tekan Enter atau Return.

```
mvn package
```

Ini menciptakan satu .jar file (JAR) di target direktori (misalnya,myapp/target). JAR ini berisi semua modul SDK yang Anda tentukan sebagai dependensi dalam file Anda. pom.xml

# Siapkan proyek Gradle

Anda dapat menggunakan [Gradle](#) untuk menyiapkan dan membangun AWS SDK for Java proyek.

Langkah awal dalam contoh berikut berasal dari [panduan Memulai Gradle](#) untuk versi 8.4. Jika Anda menggunakan versi yang berbeda, hasil Anda mungkin sedikit berbeda.

Untuk membuat aplikasi Java dengan Gradle (baris perintah)

1. Buat direktori untuk menampung proyek Anda. Dalam contoh ini, demo adalah nama direktori.
2. Di dalam demo direktori, jalankan `gradle init` perintah dan berikan nilai yang disorot dengan warna merah seperti yang ditunjukkan pada output baris perintah berikut. Untuk berjalan-jalan, kami memilih Kotlin sebagai bahasa DSL skrip build, tetapi contoh lengkap untuk Groovy juga ditampilkan di akhir topik ini.

```
> gradle init
Starting a Gradle Daemon (subsequent builds will be faster)

Select type of project to generate:
1: basic
2: application
3: library
4: Gradle plugin
Enter selection (default: basic) [1..4] 2

Select implementation language:
1: C++
2: Groovy
3: Java
4: Kotlin
5: Scala
6: Swift
Enter selection (default: Java) [1..6] 3

Generate multiple subprojects for application? (default: no) [yes, no] no
Select build script DSL:
1: Kotlin
2: Groovy
Enter selection (default: Kotlin) [1..2] <Enter>

Select test framework:
1: JUnit 4
```

```
2: TestNG
3: Spock
4: JUnit Jupiter
Enter selection (default: JUnit Jupiter) [1..4] 4

Project name (default: demo): <Enter>
Source package (default: demo): <Enter>
Enter target version of Java (min. 7) (default: 11): <Enter>
Generate build using new APIs and behavior (some features may change in the next
  minor release)? (default: no) [yes, no] <Enter>

> Task :init
To learn more about Gradle by exploring our Samples at https://docs.gradle.org/8.4/samples/sample\_building\_java\_applications.html

BUILD SUCCESSFUL in 3m 43s
2 actionable tasks: 2 executed
```

3. Setelah `init` tugas selesai, `demo` direktori berisi struktur pohon berikut. Kami melihat lebih dekat pada file build utama, `build.gradle.kts` (disorot dengan warna merah), di bagian selanjutnya.

```
### app
#   ### build.gradle.kts
#   ### src
#     ### main
#     #   ### java
#     #   #   ### demo
#     #   #       ### App.java
#     #   ### resources
#     ### test
#     #   ### java
#     #   #   ### demo
#     #   #       ### AppTest.java
#     ### resources
### gradle
#   ### wrapper
#     ### gradle-wrapper.jar
#     ### gradle-wrapper.properties
### gradlew
### gradlew.bat
### settings.gradle.kts
```

`build.gradle.ktsFile` berisi konten perancah berikut.

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

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

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

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

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

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

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


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

```
}

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

4. Gunakan file build Gradle scaffolded sebagai dasar untuk proyek Anda. AWS
  - a. Untuk mengelola dependensi SDK untuk project Gradle Anda, tambahkan Maven bill of materials (BOM) AWS SDK for Java 2.x untuk bagian file. dependencies build.gradle.kts

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

 Note

Dalam contoh file build ini, ganti 2.21.1 dengan versi terbaru SDK for Java 2.x. Temukan versi terbaru yang tersedia di repositori [pusat Maven](#).

- b. Tentukan modul SDK yang dibutuhkan aplikasi Anda di dependencies bagian ini. Sebagai contoh, berikut ini menambahkan ketergantungan pada Amazon Simple Storage Service.

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

Gradle secara otomatis menyelesaikan versi dependensi yang dideklarasikan dengan menggunakan informasi dari BOM.

Contoh berikut menunjukkan file build Gradle lengkap di DSL Kotlin dan Groovy. File build berisi dependensi untuk Amazon S3, otentikasi, logging, dan pengujian. Versi sumber dan target Java adalah versi 11.

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

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

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

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

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

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



```
    }  
  }  
  
  application {  
    // Define the main class for the application.  
    mainClass.set("demo.App")  
  }  
  
  tasks.named<Test>("test") {  
    // Use JUnit Platform for unit tests.  
    useJUnitPlatform()  
  }  
}
```

## Groovy DSL (build.gradle)

```
/*  
 * This file was generated by the Gradle 'init' task.  
 *  
 * This generated file contains a sample Java application project to get you  
 * started.  
 * For more details on building Java & JVM projects, please refer to https://docs.gradle.org/8.4/userguide/building\_java\_projects.html in the Gradle  
 * documentation.  
 */  
  
plugins {  
    // Apply the application plugin to add support for building a CLI application in  
    // Java.  
    id 'application'  
}  
  
repositories {  
    // Use Maven Central for resolving dependencies.  
    mavenCentral()  
}  
  
dependencies {  
    implementation platform('software.amazon.awssdk:bom:2.21.1')  
    implementation 'software.amazon.awssdk:s3'  
    implementation 'software.amazon.awssdk:sso'  
    implementation 'software.amazon.awssdk:ssoidc'  
    implementation platform('org.apache.logging.log4j:log4j-bom:2.20.0')  
    implementation 'org.apache.logging.log4j:log4j-slf4j2-impl'}
```

```
implementation 'org.apache.logging.log4j:log4j-1.2-api'
testImplementation platform('org.junit:junit-bom:5.10.0')
testImplementation 'org.junit.jupiter:junit-jupiter'
}

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

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

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

Untuk langkah selanjutnya, lihat panduan Memulai di situs web Gradle untuk petunjuk tentang cara [membuat dan menjalankan aplikasi Gradle](#).

## Siapkan proyek GraAlvm Native Image untuk AWS SDK for Java

Dengan versi 2.16.1 dan yang lebih baru, AWS SDK for Java menyediakan out-of-the-box dukungan untuk aplikasi GraAlvm Native Image. Gunakan pola dasar archetype-app-quickstart Maven untuk menyiapkan proyek dengan dukungan gambar asli bawaan.

### Prasyarat

- Selesaikan langkah-langkah dalam [Menyiapkan AWS SDK for Java 2.x](#).
- Instal [GraAlvm Native Image](#).

## Buat proyek menggunakan arketipe

Untuk membuat proyek Maven dengan built-in dukungan gambar asli, di terminal atau jendela command prompt, gunakan perintah berikut.

### Note

Gantikan `com.example.mynativeimageapp` dengan namespace paket lengkap aplikasi Anda. Juga gantimynativeimageapp dengan nama proyek Anda. Ini menjadi nama direktori untuk proyek Anda.

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

Perintah ini menciptakan proyek Maven dikonfigurasi dengan dependensi untuk AWS SDK for Java Amazon S3, dan klien Apache HttpClient HTTP. Ini juga mencakup ketergantungan untuk [plugin GraAlm Native Image Maven](#), sehingga Anda dapat membangun gambar asli menggunakan Maven.

Untuk menyertakan dependensi untuk yang berbeda Amazon Web Services, tetapkan nilai-Dservice parameter ke ID artefak layanan tersebut. Contohnya termasuk dynamodb, comprehend, dan pinpoint. Untuk daftar lengkap ID artefak, lihat daftar dependensi terkelola untuk [software.amazon.awssdk di Maven Central](#).

Untuk menggunakan klien HTTP asynchronous, atur-DhttpClient parameter untuk netty-nio-client. Untuk digunakan HttpURLConnectionHttpClient sebagai klien HTTP sinkron alih-alih apache-client, atur-DhttpClient parameter nya url-connection-client.

## Membangun gambar asli

Setelah Anda membuat proyek, jalankan perintah berikut dari direktori proyek Anda, misalnya, `mynativeimageapp`:

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

Ini menciptakan aplikasi gambar asli ditarget direktori, misalnya, `target/mynativeimageapp`.

# Gunakan AWS SDK for Java 2.x

Setelah menyelesaikan langkah-langkah dalam [Menyiapkan SDK](#), Anda siap untuk membuat permintaan ke AWS layanan seperti Amazon S3, DynamoDB, IAM, Amazon EC2, dan banyak lagi.

## Bekerja dengan klien layanan

### Buat klien layanan

Untuk membuat permintaan ke Layanan AWS, Anda harus terlebih dahulu membuat instance klien layanan untuk layanan tersebut dengan menggunakan metode pabrik statis, `builder()`. Metode `builder()` mengembalikan `builder` objek yang memungkinkan Anda untuk menyesuaikan klien layanan. Metode penyetel yang lancar mengembalikan `builder` objek, sehingga Anda dapat merantai panggilan metode untuk kenyamanan dan kode yang lebih mudah dibaca. Setelah Anda mengkonfigurasi properti yang Anda inginkan, panggil `build()` metode untuk membuat klien.

Sebagai contoh, cuplikan kode berikut membuat instance `Ec2Client` objek sebagai klien layanan untuk Amazon EC2.

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

#### Note

Klien layanan di SDK aman untuk utas. Untuk kinerja terbaik, perlakukan mereka sebagai benda berumur panjang. Setiap klien memiliki sumber daya kolam koneksi sendiri yang dilepaskan saat klien mengumpulkan sampah.

Objek klien layanan tidak dapat diubah, jadi Anda harus membuat klien baru untuk setiap layanan yang Anda minta, atau jika Anda ingin menggunakan konfigurasi yang berbeda untuk membuat permintaan ke layanan yang sama.

Menentukan `Region` dalam pembuat klien layanan tidak diperlukan untuk semua AWS layanan; namun, ini adalah praktik terbaik untuk mengatur Wilayah untuk panggilan API yang Anda buat dalam aplikasi Anda. Lihat [pemilihan AWS wilayah](#) untuk informasi lebih lanjut.

## Konfigurasi klien default

Pembangun klien memiliki metode pabrik lain bernama `create()`. Metode ini menciptakan klien layanan dengan konfigurasi default. Ini menggunakan rantai penyedia default untuk memuat kredensial dan file. Wilayah AWS Jika kredensial atau Wilayah tidak dapat ditentukan dari lingkungan tempat aplikasi berjalan, panggilan ke `create` gagal. Lihat [Menggunakan kredensial](#) dan [pemilihan Wilayah](#) untuk informasi selengkapnya tentang cara SDK menentukan kredensial dan Wilayah yang akan digunakan.

Misalnya, cuplikan kode berikut membuat instance `DynamoDbClient` objek sebagai klien layanan untuk Amazon DynamoDB:

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

## Konfigurasi klien layanan

Untuk menyesuaikan konfigurasi klien layanan, gunakan setter pada metode `builder()` pabrik. Untuk kenyamanan dan untuk membuat kode yang lebih mudah dibaca, rantai metode untuk mengatur beberapa opsi konfigurasi.

Contoh berikut menunjukkan `S3Client` yang dikonfigurasi dengan beberapa pengaturan kustom.

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

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

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

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

## Membuat permintaan

Gunakan klien layanan untuk membuat permintaan ke yang sesuai Layanan AWS.

Misalnya, cuplikan kode ini menunjukkan cara membuat `RunInstancesRequest` objek untuk membuat instance Amazon EC2 baru:

```
// Create the request by using the fluid setter methods of the request builder.
RunInstancesRequest runInstancesRequest = RunInstancesRequest.builder()
    .imageId(amiId)
    .instanceType(InstanceType.T1_MICRO)
    .maxCount(1)
    .minCount(1)
    .build();

// Use the configured request with the service client.
RunInstancesResponse response = ec2Client.runInstances(runInstancesRequest);
```

Alih-alih membuat permintaan dan meneruskan instance, SDK menyediakan builder yang dapat Anda gunakan untuk membuat permintaan. Dengan pembangun Anda dapat menggunakan ekspresi lambda Java untuk membuat permintaan 'in-baris'.

Contoh berikut menulis ulang contoh sebelumnya dengan menggunakan versi `runInstances` [metode yang menggunakan pembangun](#) untuk membuat permintaan.

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

## Menangani tanggapan

Anda menggunakan penanganan respons untuk memproses respons kembali dari Layanan AWS

Misalnya, cuplikan kode ini menunjukkan cara membuat `RunInstancesResponse` objek untuk menangani respons dari Amazon EC2 dengan mencetak instance baru `instanceId` dari permintaan di atas:

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

## Tutup klien layanan

Sebagai praktik terbaik, Anda harus menggunakan klien layanan untuk beberapa panggilan layanan API selama masa pakai aplikasi. Namun, jika Anda membutuhkan klien layanan untuk penggunaan satu kali atau tidak lagi membutuhkan klien layanan, tutuplah.

Panggil `close()` metode ketika klien layanan tidak lagi diperlukan untuk membebaskan sumber daya.

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

Jika Anda memerlukan klien layanan untuk penggunaan satu kali, Anda dapat membuat instance klien layanan sebagai sumber daya dalam pernyataan `-with-resources`. `try` Klien layanan mengimplementasikan [Autoclosable](#) antarmuka, sehingga JDK secara otomatis memanggil `close()` metode di akhir pernyataan.

Contoh berikut menunjukkan cara menggunakan klien layanan untuk panggilan satu kali. `StsClient` yang memanggil AWS Security Token Service ditutup setelah mengembalikan ID akun.

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

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

## Menangani pengecualian

SDK menggunakan pengecualian runtime (atau tidak dicentang), memberi Anda kontrol halus atas penanganan kesalahan dan memastikan bahwa penanganan pengecualian akan diskalakan dengan aplikasi Anda.

An [SdkServiceException](#), atau salah satu sub-kelasnya, adalah bentuk pengecualian yang paling umum yang akan dilemparkan SDK. Pengecualian ini mewakili tanggapan dari AWS layanan. Anda



juga dapat menangani [SdkClientException](#), yang terjadi ketika ada masalah di sisi klien (yaitu, dalam pengembangan atau lingkungan aplikasi Anda), seperti kegagalan koneksi jaringan.

Cuplikan kode ini menunjukkan salah satu cara untuk menangani pengecualian layanan saat Anda mengunggah file ke Amazon S3. Kode contoh menangkap pengecualian klien dan server, mencatat detailnya, dan ada aplikasi.

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

try {

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

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

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

Lihat [Menangani pengecualian](#) untuk informasi selengkapnya.

## Gunakan pelayan

Beberapa permintaan membutuhkan waktu untuk diproses, seperti membuat tabel baru DynamoDB atau membuat Amazon S3 bucket baru. Untuk memastikan sumber daya siap sebelum kode Anda terus berjalan, gunakan Pelayan.

Misalnya, cuplikan kode ini membuat tabel baru (“myTable”) di DynamoDB, menunggu tabel berada dalam ACTIVE status, dan kemudian mencetak respons:

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

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

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

Lihat [Menggunakan pelayan](#) untuk informasi lebih lanjut.

## Klien HTTP

Anda dapat mengubah konfigurasi default untuk klien HTTP dalam aplikasi yang Anda buat dengan AWS SDK for Java. Untuk informasi tentang cara mengkonfigurasi klien dan pengaturan HTTP, lihat [konfigurasi HTTP](#).

## Percobaan ulang

Anda dapat mengubah pengaturan default untuk percobaan ulang di klien layanan Anda, termasuk mode coba lagi dan strategi mundur. Untuk informasi selengkapnya, lihat [RetryPolicy](#) kelas di Referensi AWS SDK for Java API.

Untuk informasi selengkapnya tentang percobaan ulang di AWS layanan, lihat [Error retries dan exponential](#) backoff in. AWS

## Timeout

Anda dapat mengonfigurasi batas waktu untuk setiap klien layanan Anda menggunakan `apiCallTimeout` dan `apiCallAttemptTimeout` setter. `apiCallTimeout` pengaturan adalah jumlah waktu untuk memungkinkan klien menyelesaikan eksekusi panggilan API. `apiCallAttemptTimeout` pengaturan adalah jumlah waktu untuk menunggu permintaan HTTP selesai sebelum menyerah.

Untuk informasi selengkapnya, lihat [apiCallTimeout](#) dan [apiCallAttemptTimeout](#) di Referensi AWS SDK for Java API.

## Pencegat eksekusi

Anda dapat menulis kode yang mencegah eksekusi permintaan dan tanggapan API Anda di berbagai bagian siklus hidup permintaan/respons. Ini memungkinkan Anda untuk mempublikasikan metrik, memodifikasi permintaan dalam penerbangan, men-debug pemrosesan permintaan, melihat pengecualian, dan banyak lagi. Untuk informasi selengkapnya, lihat [ExecutionInterceptorantarmuka](#) di Referensi AWS SDK for Java API.

## Informasi tambahan

- Untuk contoh lengkap cuplikan kode di atas, lihat [Bekerja dengan Amazon DynamoDB](#), [Bekerja dengan Amazon EC2](#), dan [Bekerja dengan Amazon S3](#)

## Memberikan kredensi sementara ke SDK

Sebelum membuat permintaan untuk Amazon Web Services menggunakan AWS SDK for Java 2.x, SDK secara kriptografis menandatangani kredensial sementara yang dikeluarkan oleh AWS. Untuk mengakses kredensial sementara, SDK mengambil nilai konfigurasi dengan memeriksa beberapa lokasi.

Topik ini membahas beberapa cara Anda mengaktifkan SDK untuk mengakses kredensial sementara.

Topik

- [Konfigurasi akses ke kredensial sementara](#)
- [Rantai penyedia kredensi default](#)
- [Menggunakan penyedia kredensial tertentu atau rantai penyedia](#)
- [Gunakan profil](#)
- [Memuat kredensi sementara dari proses eksternal](#)
- [Menyediakan kredensi sementara dalam kode](#)
- [Baca kredensi peran IAM di Amazon EC2](#)

## Konfigurasi akses ke kredensial sementara

Untuk meningkatkan keamanan, AWS merekomendasikan agar Anda mengonfigurasi SDK for Java agar [menggunakan kredensial sementara](#), bukan kredensial yang berumur panjang. Kredensial sementara terdiri dari kunci akses (id kunci akses dan kunci akses rahasia) dan token sesi. Kami menyarankan Anda [mengonfigurasi SDK](#) untuk mendapatkan kredensial sementara secara otomatis, karena proses penyegaran token otomatis. Namun, Anda dapat [memberikan SDK kredensial sementara](#) secara langsung.

### Konfigurasi Pusat Identitas IAM

Saat Anda mengonfigurasi SDK untuk menggunakan akses masuk tunggal Pusat Identitas IAM seperti yang dijelaskan [???](#) dalam panduan ini, SDK secara otomatis menggunakan kredensial sementara.

SDK menggunakan token akses Pusat Identitas IAM untuk mendapatkan akses ke peran IAM yang dikonfigurasi dengan `sso_role_name` pengaturan di file `Anda.config` SDK mengasumsikan peran IAM ini dan mengambil kredensial sementara untuk digunakan untuk permintaan. Layanan AWS

Untuk detail selengkapnya tentang cara SDK mendapatkan kredensial sementara dari konfigurasi, lihat bagian [Memahami autentikasi Pusat Identitas IAM](#) dari AWS SDK dan Panduan Referensi Alat.

### Ambil dari portal AWS akses

Sebagai alternatif untuk konfigurasi akses masuk tunggal Pusat Identitas IAM, Anda dapat menyalin dan menggunakan kredensial sementara yang tersedia di portal akses. AWS Anda dapat menggunakan kredensial sementara di profil atau menggunakannya sebagai nilai untuk properti sistem dan variabel lingkungan.

### Menyiapkan file kredensial lokal untuk kredensial sementara

1. [Buat file kredensial bersama](#)
2. Dalam file kredensial, rekatkan teks placeholder berikut hingga Anda menempelkan kredensial sementara yang berfungsi.

```
[default]
aws_access_key_id=<value from AWS access portal>
aws_secret_access_key=<value from AWS access portal>
aws_session_token=<value from AWS access portal>
```



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

## Urutan pengambilan pengaturan kredensi

Rantai penyedia kredensial default SDK for Java 2.x mencari konfigurasi di lingkungan Anda menggunakan urutan yang telah ditentukan sebelumnya.

### 1. Properti sistem Java

- SDK menggunakan [SystemPropertyCredentialsProvider](#) kelas untuk memuat kredensi sementara dari properti sistem `aws.accessKeyId`, `aws.secretAccessKey`, dan `aws.sessionToken` Java.

#### Note

Untuk informasi tentang cara mengatur properti sistem Java, lihat tutorial [System Properties](#) di situs web resmi Java Tutorial.

### 2. Variabel-variabel lingkungan

- SDK menggunakan [EnvironmentVariableCredentialsProvider](#) kelas untuk memuat kredensial sementara dari variabel `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, dan `AWS_SESSION_TOKEN` lingkungan.

### 3. Token identitas web dari AWS Security Token Service

- SDK menggunakan [WebIdentityTokenFileCredentialsProvider](#) kelas untuk memuat kredensial sementara dari properti sistem Java atau variabel lingkungan.

### 4. Yang dibagikan `credentials` dan `config` file

- SDK menggunakan pengaturan masuk tunggal IAM Identity Center atau kredensial sementara dari [default] profil di file bersama dan file. [ProfileCredentialsProvider](#) `credentialsconfig`

Panduan Referensi AWS SDK dan Alat memiliki [informasi terperinci](#) tentang cara kerja SDK for Java dengan token masuk tunggal IAM Identity Center untuk mendapatkan kredensial sementara yang digunakan SDK untuk memanggil. Layanan AWS

**Note**

`configFile` `credentials` dan dibagikan oleh berbagai AWS SDK dan Alat. Untuk informasi selengkapnya, lihat [File.aws/credentials](#) dan [.aws/config](#) di [SDK dan Tools Reference Guide](#). AWS

## 5. Amazon EC2 kredensi kontainer

- SDK menggunakan [ContainerCredentialsProvider](#) kelas untuk memuat kredensial sementara dari variabel lingkungan `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` sistem.

## 6. Amazon EC2 kredensi yang disediakan peran IAM misalnya

- SDK menggunakan [InstanceProfileCredentialsProvider](#) kelas untuk memuat kredensial sementara dari layanan metadata. Amazon EC2

## Menggunakan penyedia kredensial tertentu atau rantai penyedia

Sebagai alternatif dari rantai penyedia kredensial default, Anda dapat menentukan penyedia kredensial mana yang harus digunakan SDK. Ketika Anda menyediakan penyedia kredensi tertentu, SDK melewati proses memeriksa berbagai lokasi, yang sedikit mengurangi waktu untuk membuat klien layanan.

Misalnya, jika Anda menyetel konfigurasi default menggunakan variabel lingkungan, berikan [EnvironmentVariableCredentialsProvider](#) objek ke `credentialsProvider` metode pada pembuat klien layanan, seperti pada cuplikan kode berikut.

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

Untuk daftar lengkap penyedia kredensial dan rantai penyedia, lihat Semua Kelas Penerapan yang Dikenal di [AwsCredentialsProvider](#).

**Note**

Anda dapat menggunakan penyedia kredensial atau rantai penyedia Anda sendiri dengan mengimplementasikan antarmuka `AwsCredentialsProvider`

## Gunakan profil

Menggunakan `credentials` file bersama `config` dan, Anda dapat mengatur beberapa profil. Hal ini memungkinkan aplikasi Anda untuk menggunakan beberapa set konfigurasi kredensial. [default]Profil tersebut telah disebutkan sebelumnya. SDK menggunakan [ProfileCredentialsProvider](#) kelas untuk memuat pengaturan dari profil yang ditentukan dalam `credentials` file bersama.

Cuplikan kode berikut menunjukkan cara membangun klien layanan yang menggunakan pengaturan yang didefinisikan sebagai bagian dari profil bernama `my_profile`

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

## Tetapkan profil yang berbeda sebagai default

Untuk menetapkan profil selain [default] profil sebagai default untuk aplikasi Anda, setel variabel `AWS_PROFILE` lingkungan ke nama profil kustom Anda.

Untuk mengatur variabel ini di Linux, macOS, atau Unix, gunakan: `export`

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

Untuk menetapkan variabel ini di Windows, gunakan `set`:

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

Atau, atur properti sistem `aws.profile` Java ke nama profil.



## Muat ulang kredensi profil

Anda dapat mengonfigurasi penyedia kredensial apa pun yang memiliki `profileFile()` metode pada pembuatnya untuk memuat ulang kredensial profil. Kelas profil kredensial ini adalah: `ProfileCredentialsProvider`, `DefaultCredentialsProvider`, dan `InstanceProfileCredentialsProvider` `ProfileTokenProvider`.

### Note

Pemuatan ulang kredensial profil hanya berfungsi dengan pengaturan berikut di file profil: `aws_access_key_id`, `aws_secret_access_key`, dan `aws_session_token`. Pengaturan seperti `region`, `sso_session`, `sso_account_id`, dan `source_profile` diabaikan.

Untuk mengonfigurasi penyedia kredensial yang didukung untuk memuat ulang setelan profil, berikan instance [ProfileFileSupplier](#) ke metode `profileFile()` pembangun. Contoh kode berikut menunjukkan `ProfileCredentialsProvider` yang memuat ulang pengaturan kredensial dari profil. [default]

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

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

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

Ketika `ProfileCredentialsProvider.resolveCredentials()` dipanggil, SDK for Java memuat ulang pengaturan. `ProfileFileSupplier.defaultSupplier()` adalah salah satu dari [beberapa implementasi kenyamanan](#) yang `ProfileFileSupplier` disediakan oleh SDK. Jika kasus penggunaan Anda membutuhkan, Anda dapat memberikan implementasi Anda sendiri.

Contoh berikut menunjukkan penggunaan metode `ProfileFileSupplier.reloadWhenModified()` kenyamanan. `reloadWhenModified()` mengambil `Path` parameter, yang memberi Anda fleksibilitas dalam menunjuk file sumber untuk konfigurasi daripada lokasi standar `~/.aws/credentials` (atau `config`).

Pengaturan akan dimuat ulang ketika `resolveCredentials()` dipanggil hanya jika SDK menentukan konten file telah dimodifikasi.

```
Path credentialsFilePath = ...

ProfileCredentialsProvider provider = ProfileCredentialsProvider
    .builder()
    .profileFile(ProfileFileSupplier.reloadWhenModified(credentialsFilePath,
ProfileFile.Type.CREDENTIALS))
    .profileName("my-profile")
    .build();
/*
   A service client configured with the provider instance calls
   provider.resolveCredential()
   before each request.
*/
```

`ProfileFileSupplier.aggregate()` Metode ini menggabungkan isi dari beberapa file konfigurasi. Anda memutuskan apakah file dimuat ulang per panggilan ke `resolveCredentials()` atau pengaturan file diperbaiki pada saat pertama kali dibaca.

Contoh berikut menunjukkan `DefaultCredentialsProvider` yang menggabungkan pengaturan dua file yang berisi pengaturan profil. SDK memuat ulang pengaturan dalam file yang ditunjuk oleh `credentialsFilePath` variabel setiap kali `resolveCredentials()` dipanggil dan pengaturan telah berubah. Pengaturan dari `profileFile` objek tetap sama.

```
Path credentialsFilePath = ...;
ProfileFile profileFile = ...;

DefaultCredentialsProvider provider = DefaultCredentialsProvider
    .builder()
    .profileFile(ProfileFileSupplier.aggregate(
        ProfileFileSupplier.reloadWhenModified(credentialsFilePath,
ProfileFile.Type.CREDENTIALS),
        ProfileFileSupplier.fixedProfileFile(profileFile)))
```

```
        .profileName("my-profile")
        .build();
/*
   A service client configured with the provider instance calls
   provider.resolveCredential()
   before each request.
*/
```

## Memuat kredensi sementara dari proses eksternal

### Warning

Berikut ini menjelaskan metode sumber kredensial sementara dari proses eksternal. Ini berpotensi berbahaya, jadi lanjutkan dengan hati-hati. Penyedia kredensi lainnya harus lebih disukai jika memungkinkan. Jika menggunakan opsi ini, Anda harus memastikan bahwa config file tersebut dikunci sebanyak mungkin menggunakan praktik terbaik keamanan untuk sistem operasi Anda.

Pastikan alat kredensial kustom Anda tidak menulis informasi rahasia apa pun. StdErr SDK dan AWS CLI dapat menangkap dan mencatat informasi tersebut, berpotensi mengeksposnya kepada pengguna yang tidak sah.

Dengan SDK for Java 2.x, Anda dapat memperoleh kredensial sementara dari proses eksternal untuk kasus penggunaan kustom. Ada dua cara untuk mengkonfigurasi fungsi ini.

### Gunakan **credential\_process** pengaturan

Jika Anda memiliki metode yang menyediakan kredensi sementara, Anda dapat mengintegrasikannya dengan menambahkan `credential_process` pengaturan sebagai bagian dari definisi profil dalam file. `config` Nilai yang Anda tentukan harus menggunakan path lengkap ke file perintah. Jika jalur file berisi spasi apa pun, Anda harus mengelilinginya dengan tanda kutip.

SDK memanggil perintah persis seperti yang diberikan dan kemudian membaca data JSON dari `stdout`

Contoh berikut menunjukkan penggunaan pengaturan ini untuk jalur file tanpa spasi dan jalur file dengan spasi.

## Linux/macOS

### Tidak ada spasi di jalur file

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

### Spasi di jalur file

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

## Windows

### Tidak ada spasi di jalur file

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

### Spasi di jalur file

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

Cuplikan kode berikut menunjukkan cara membangun klien layanan yang menggunakan kredensial sementara yang didefinisikan sebagai bagian dari profil bernama. `process-credential-profile`

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

Untuk informasi terperinci tentang penggunaan proses eksternal sebagai sumber kredensial sementara, lihat [bagian kredensial proses di AWS SDK dan Panduan Referensi Alat](#).

## Gunakan `ProcessCredentialsProvider`

Sebagai alternatif untuk menggunakan pengaturan dalam config file, Anda dapat menggunakan SDK [ProcessCredentialsProvider](#) untuk memuat kredensial sementara menggunakan Java.

Contoh berikut menunjukkan berbagai versi cara menentukan proses eksternal menggunakan `ProcessCredentialsProvider` dan mengkonfigurasi klien layanan yang menggunakan kredensi sementara.

### Linux/macOS

Tidak ada spasi di jalur file

```
ProcessCredentialsProvider credentials =
    ProcessCredentialsProvider
        .builder()
        .command("/path/to/credentials.sh optional_param1 optional_param2")
        .build();

S3Client s3 = S3Client.builder()
    .region(Region.US_WEST_2)
    .credentialsProvider(credentials)
    .build();
```

Spasi di jalur file

```
ProcessCredentialsProvider credentials =
    ProcessCredentialsProvider
        .builder()
        .command("/path\\ with\\ spaces\\ to/credentials.sh optional_param1
optional_param2")
        .build();

S3Client s3 = S3Client.builder()
    .region(Region.US_WEST_2)
    .credentialsProvider(credentials)
    .build();
```

## Windows

### Tidak ada spasi di jalur file

```
ProcessCredentialsProvider credentials =
    ProcessCredentialsProvider
        .builder()
        .command("C:\\Path\\To\\credentials.exe optional_param1 optional_param2")
        .build();

S3Client s3 = S3Client.builder()
    .region(Region.US_WEST_2)
    .credentialsProvider(credentials)
    .build();
```

### Spasi di jalur file

```
ProcessCredentialsProvider credentials =
    ProcessCredentialsProvider
        .builder()
        .command("\"C:\\Path\\With Spaces To\\credentials.exe\" optional_param1
optional_param2")
        .build();

S3Client s3 = S3Client.builder()
    .region(Region.US_WEST_2)
    .credentialsProvider(credentials)
    .build();
```

## Menyediakan kredensi sementara dalam kode

Jika rantai kredensi default atau penyedia atau rantai penyedia khusus atau khusus tidak berfungsi untuk aplikasi Anda, Anda dapat memberikan kredensi sementara secara langsung dalam kode. Ini bisa berupa kredensi [peran IAM seperti yang dijelaskan di atas atau kredensial](#) sementara yang diambil dari (). AWS Security Token Service AWS STS Jika Anda mengambil kredensial sementara menggunakan AWS STS, berikan kepada Layanan AWS klien seperti yang ditunjukkan dalam contoh kode berikut.

1. Asumsikan peran dengan menelepon `stsClient.assumeRole()`.
2. Buat [StaticCredentialsProvider](#) objek dan suplai dengan `AwsSessionCredentials` objek.

### 3. Konfigurasi pembuat klien layanan dengan `StaticCredentialsProvider` dan bangun klien.

Contoh berikut membuat klien layanan Amazon S3 menggunakan kredensial sementara yang dikembalikan oleh AWS STS untuk peran yang diasumsikan IAM.

```
// The AWS IAM Identity Center identity (user) who executes this method does not
// have permission to list buckets.
// The identity is configured in the [default] profile.
public static void assumeRole(String roleArn, String roleSessionName) {
    // The IAM role represented by the 'roleArn' parameter can be assumed by
    // identities in two different accounts
    // and the role permits the user to only list buckets.

    // The SDK's default credentials provider chain will find the single sign-on
    // settings in the [default] profile.
    // The identity configured with the [default] profile needs permission to call
    // AssumeRole on the STS service.
    try {
        Credentials tempRoleCredentials;
        try (StsClient stsClient = StsClient.create()) {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                .roleArn(roleArn)
                .roleSessionName(roleSessionName)
                .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            tempRoleCredentials = roleResponse.credentials();
        }
        // Use the following temporary credential items for the S3 client.
        String key = tempRoleCredentials.accessKeyId();
        String secKey = tempRoleCredentials.secretAccessKey();
        String secToken = tempRoleCredentials.sessionToken();

        // List all buckets in the account associated with the assumed role
        // by using the temporary credentials retrieved by invoking
        stsClient.assumeRole().
        StaticCredentialsProvider staticCredentialsProvider =
        StaticCredentialsProvider.create(
            AwsSessionCredentials.create(key, secKey, secToken));
        try (S3Client s3 = S3Client.builder()
            .credentialsProvider(staticCredentialsProvider)
```

```
        .build()) {
        List<Bucket> buckets = s3.listBuckets().buckets();
        for (Bucket bucket : buckets) {
            System.out.println("bucket name: " + bucket.name());
        }
    }
} catch (StsException | S3Exception e) {
    logger.error(e.getMessage());
    System.exit(1);
}
}
```

## Set izin

Set izin berikut yang didefinisikan dalam AWS IAM Identity Center memungkinkan identitas (pengguna) untuk melakukan dua operasi berikut

1. `GetObject` Pengoperasian Layanan Penyimpanan Sederhana Amazon.
2. `AssumeRole` Pengoperasian AWS Security Token Service.

Tanpa mengasumsikan peran, `s3.listBuckets()` metode yang ditunjukkan dalam contoh akan gagal.

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



## Asumsi peran

### Kebijakan izin peran yang diasumsikan

Kebijakan izin berikut dilampirkan pada peran yang diasumsikan dalam contoh sebelumnya. Polilitas izin ini memungkinkan kemampuan untuk mencantumkan semua bucket di akun yang sama dengan peran.

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

### Kebijakan kepercayaan peran yang diasumsikan

Kebijakan kepercayaan berikut dilampirkan pada peran yang diasumsikan dalam contoh sebelumnya. Kebijakan ini memungkinkan peran diasumsikan oleh identitas (pengguna) dalam dua akun.

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

```
]
}
```

## Baca kredensi peran IAM di Amazon EC2

Anda dapat menggunakan peran IAM untuk mengelola kredensial sementara untuk aplikasi yang berjalan pada instans EC2 dan membuat AWS CLI atau permintaan API. AWS Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan AWS peran ke instans EC2 dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat [Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di instans Amazon EC2](#) dalam Panduan Pengguna IAM.

Topik ini memberikan informasi tentang cara mengatur aplikasi Java Anda agar berjalan pada instans EC2 dan mengaktifkan SDK for Java untuk IAM memperoleh kredensi peran.

## Memperoleh kredensi peran IAM dari lingkungan

Jika aplikasi Anda membuat klien AWS layanan menggunakan metode (atau `create builder().build()` metode), SDK for Java menggunakan rantai penyedia kredensial default. Rantai penyedia kredensial default mencari lingkungan eksekusi untuk elemen konfigurasi yang dapat diperdagangkan SDK untuk kredensial sementara. [the section called “Rantai penyedia kredensial default”](#) Bagian ini menjelaskan proses pencarian lengkap.

Langkah terakhir dalam rantai penyedia default hanya tersedia ketika aplikasi Anda berjalan pada sebuah Amazon EC2 instance. Pada langkah ini, SDK menggunakan `InstanceProfileCredentialsProvider` untuk membaca peran IAM yang ditentukan dalam profil instans EC2. SDK kemudian memperoleh kredensi sementara untuk peran IAM tersebut.

Meskipun kredensi ini bersifat sementara dan pada akhirnya akan kedaluwarsa, `InstanceProfileCredentialsProvider` secara berkala menyegarkannya untuk Anda sehingga mereka terus mengizinkan akses ke AWS

## Memperoleh kredensi peran IAM secara terprogram

Sebagai alternatif dari rantai penyedia kredensial default yang pada akhirnya menggunakan `InstanceProfileCredentialsProvider` on EC2, Anda dapat mengonfigurasi klien layanan

secara eksplisit dengan file. `InstanceProfileCredentialsProvider` Pendekatan ini ditunjukkan dalam cuplikan berikut.

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

## Dapatkan kredensi peran IAM dengan aman

Secara default, instans EC2 menjalankan [IMDS](#) (Layanan Metadata Instans) yang memungkinkan SDK mengakses informasi seperti `InstanceProfileCredentialsProvider` peran IAM yang telah dikonfigurasi. Instans EC2 menjalankan dua versi IMDS secara default:

- Layanan Metadata Instans Versi 1 (IMDSv1) – metode permintaan/tanggapan
- Layanan Metadata Instans Versi 2 (IMDSv2) - metode berorientasi sesi

[IMDSv2 adalah pendekatan yang lebih aman daripada IMDSv1.](#)

Secara default, Java SDK pertama kali mencoba IMDSv2 untuk mendapatkan peran IAM, tetapi jika gagal, ia mencoba IMDSv1. Namun, karena IMDSv1 kurang aman, AWS merekomendasikan penggunaan IMDSv2 saja dan untuk menonaktifkan SDK dari mencoba IMDSv1.

Untuk menggunakan pendekatan yang lebih aman, nonaktifkan SDK dari penggunaan IMDSv1 dengan memberikan salah satu pengaturan berikut dengan nilai `true`

- Variabel lingkungan: `AWS_EC2_METADATA_V1_DISABLED`
- Properti sistem JVM: `aws.disableEc2MetadataV1`
- Pengaturan file konfigurasi bersama: `ec2_metadata_v1_disabled`

Dengan salah satu pengaturan ini disetel ke `true`, SDK tidak memuat kredensi peran IMDS dengan menggunakan IMDSv1 jika panggilan IMDSv2 awal gagal.

## Gunakan Wilayah AWS

Wilayah AWS memungkinkan klien layanan untuk mengakses Layanan AWS yang secara fisik berada di wilayah geografis tertentu.

## Konfigurasi secara eksplisit Wilayah AWS

[Untuk secara eksplisit menetapkan Region, kami sarankan Anda menggunakan konstanta yang didefinisikan dalam kelas Region.](#) Ini adalah enumerasi dari semua wilayah yang tersedia untuk umum.

Untuk membuat klien dengan Region yang disebutkan dari kelas, gunakan metode pembuat klien. `region`

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

Jika Region yang ingin Anda gunakan bukan salah satu enumerasi di `Region` kelas, Anda dapat membuat Region baru dengan menggunakan metode statis. `of` Metode ini memungkinkan Anda mengakses Wilayah baru tanpa memutakhirkan SDK.

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

### Note

Setelah Anda membangun klien dengan pembangun, itu tidak dapat diubah dan Wilayah AWS tidak dapat diubah. Jika Anda perlu bekerja dengan beberapa Wilayah AWS untuk layanan yang sama, Anda harus membuat beberapa klien—satu per Wilayah.

## Biarkan SDK secara otomatis menentukan Wilayah dari lingkungan

Ketika kode Anda berjalan pada Amazon EC2 atau AWS Lambda, Anda mungkin ingin mengkonfigurasi klien untuk menggunakan Wilayah AWS yang sama dengan kode Anda berjalan. Ini memisahkan kode Anda dari lingkungan tempat ia berjalan dan membuatnya lebih mudah untuk menerapkan aplikasi Anda ke beberapa Wilayah AWS untuk latensi atau redundansi yang lebih rendah.

Untuk menggunakan rantai penyedia kredensial/wilayah default untuk menentukan Wilayah dari lingkungan, gunakan metode pembuat klien. `create`

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

Jika Anda tidak menyetel secara eksplisit menggunakan `region` metode ini, SDK akan berkonsultasi Wilayah AWS dengan rantai penyedia wilayah default untuk menentukan Wilayah yang akan digunakan.

## Memahami rantai penyedia wilayah default

SDK mengambil langkah-langkah berikut untuk mencari: Wilayah AWS

1. Setiap Wilayah eksplisit yang disetel dengan menggunakan `region` pada pembuat itu sendiri lebih diutamakan daripada yang lainnya.
2. Variabel `AWS_REGION` lingkungan diperiksa. Jika disetel, Wilayah itu digunakan untuk mengkonfigurasi klien.

### Note

Lambda Wadah menetapkan variabel lingkungan ini.

3. SDK memeriksa file konfigurasi AWS bersama dan file kredensial bersama (biasanya terletak di `~/.aws/config` dan `~/.aws/credentials`). Jika `region` properti ada, SDK menggunakannya.
  - Jika SDK menemukan `region` properti di kedua file untuk profil yang sama (termasuk default profil), SDK akan menggunakan nilai dalam file kredensial bersama.
  - Variabel `AWS_CONFIG_FILE` lingkungan dapat digunakan untuk menyesuaikan lokasi file konfigurasi bersama.
  - Variabel `AWS_PROFILE` lingkungan atau properti `aws.profile` sistem dapat digunakan untuk menentukan profil yang dimuat SDK.
4. SDK mencoba menggunakan layanan metadata Amazon EC2 instance (IMDS) untuk menentukan Wilayah instance yang sedang berjalan. Amazon EC2
  - Untuk keamanan yang lebih besar, Anda harus menonaktifkan SDK dari mencoba menggunakan IMDS versi 1. Anda menggunakan pengaturan yang sama untuk menonaktifkan versi 1 yang dijelaskan di [the section called “Aman”](#) bagian.
5. Jika SDK masih belum menemukan Wilayah pada titik ini, pembuatan klien gagal dengan pengecualian.

Saat mengembangkan AWS aplikasi, pendekatan umum adalah dengan menggunakan file konfigurasi bersama (dijelaskan dalam [urutan pengambilan Kredensial](#)) untuk mengatur Wilayah untuk pengembangan lokal, dan mengandalkan rantai penyedia wilayah default untuk menentukan Wilayah saat aplikasi berjalan pada AWS infrastruktur. Ini sangat menyederhanakan pembuatan klien dan membuat aplikasi Anda portabel.

## Periksa ketersediaan layanan di suatu Wilayah

Untuk melihat apakah tertentu Layanan AWS tersedia di Wilayah, gunakan `region` metode `serviceMetadata` dan pada klien layanan.

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

Lihat dokumentasi kelas [Region](#) untuk Wilayah AWS Anda dapat menentukan, dan menggunakan awalan endpoint dari layanan untuk query.

## Pilih titik akhir tertentu

Dalam situasi tertentu—seperti untuk menguji fitur pratinjau layanan sebelum fitur lulus ke ketersediaan umum—Anda mungkin perlu menentukan titik akhir tertentu di Wilayah. Dalam situasi ini, klien layanan dapat dikonfigurasi dengan memanggil `endpointOverride` metode.

Misalnya, untuk mengonfigurasi Amazon EC2 klien agar menggunakan Wilayah Eropa (Irlandia) dengan titik akhir tertentu, gunakan kode berikut.

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

Lihat [Wilayah dan Titik Akhir](#) untuk daftar wilayah saat ini dan titik akhir yang sesuai untuk semua AWS layanan.

## Kurangi waktu startup SDK untuk AWS Lambda

Salah satu tujuannya AWS SDK for Java 2.x adalah untuk mengurangi latensi startup untuk AWS Lambda fungsi. SDK berisi perubahan yang mengurangi waktu startup, yang dibahas di akhir topik ini.

Pertama, topik ini berfokus pada perubahan yang dapat Anda lakukan untuk mengurangi waktu mulai dingin. Ini termasuk membuat perubahan dalam struktur kode Anda dan dalam konfigurasi klien layanan.

## Gunakan SDK `URLConnectionHttpClient`

Untuk skenario sinkron, SDK for Java 2.x menawarkan [URLConnectionHttpClient](#) kelas, yang didasarkan pada kelas klien HTTP JDK. Karena `URLConnectionHttpClient` didasarkan pada kelas yang sudah ada di classpath, tidak ada dependensi tambahan untuk dimuat.

Untuk informasi tentang menambahkan `URLConnectionHttpClient` ke proyek Lambda Anda dan mengonfigurasi penggunaannya, lihat. [Konfigurasi klien HTTP berbasis URLConnection](#)

### Note

Ada beberapa keterbatasan fitur `URLConnectionHttpClient` dibandingkan dengan SDK. [ApacheHttpClient](#) `ApacheHttpClient` ini adalah klien HTTP asinkron default di SDK. Misalnya, `URLConnectionHttpClient` tidak mendukung metode HTTP PATCH. Beberapa operasi AWS API memerlukan permintaan PATCH. Nama-nama operasi itu biasanya dimulai dengan `Update*`. Berikut ini adalah beberapa contoh.

- [Beberapa Update\\* operasi](#) di AWS Security Hub API dan juga [BatchUpdateFindings](#) operasi
- Semua [Update\\* operasi](#) API Amazon API Gateway
- [Beberapa Update\\* operasi](#) di Amazon WorkDocs API

Jika Anda mungkin menggunakan `URLConnectionHttpClient`, pertama-tama lihat Referensi API untuk Layanan AWS yang Anda gunakan. Periksa untuk melihat apakah operasi yang Anda butuhkan menggunakan operasi PATCH.

## Gunakan SDK `AwsCrtAsyncHttpClient`

[AwsCrtAsyncHttpClient](#) ini adalah mitra asinkron untuk mengurangi waktu startup Lambda di SDK.

`AwsCrtAsyncHttpClient` ini adalah klien HTTP asinkron, non-pemblokiran. Ini dibangun di atas binding Java dari AWS Common Runtime, yang ditulis dalam bahasa pemrograman C. Di antara tujuan dalam pengembangan AWS Common Runtime adalah kinerja yang cepat.

Bagian panduan tentang [mengonfigurasi klien HTTP](#) ini memiliki informasi tentang menambahkan `AwsCrtAsyncHttpClient` ke proyek Lambda Anda dan mengonfigurasi penggunaannya.

## Hapus dependensi klien HTTP yang tidak digunakan

Seiring dengan penggunaan eksplisit `URLConnectionHttpClient` or `AwsCrtAsyncHttpClient`, Anda dapat menghapus klien HTTP lain yang dibawa SDK secara default. Waktu startup Lambda berkurang ketika lebih sedikit pustaka yang perlu dimuat, jadi Anda harus menghapus artefak yang tidak terpakai yang perlu dimuat JVM.

Cuplikan berikut dari `pom.xml` file Maven menunjukkan pengecualian klien HTTP berbasis Apache dan klien HTTP berbasis Netty. (Klien ini tidak diperlukan saat Anda menggunakan `URLConnectionHttpClient`.) Contoh ini mengecualikan artefak klien HTTP dari ketergantungan klien S3 dan menambahkan `url-connection-client` artefak, yang membawa kelas `URLConnectionHttpClient`

```
<project>
  <properties>
    <aws.java.sdk.version>2.17.290</aws.java.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.java.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>url-connection-client</artifactId>
    </dependency>
  </dependencies>
</project>
```



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

Jika Anda menggunakan `AwsCrtAsyncHttpClient`, ganti dependensi pada `url-connection-client` dependensi pada `aws-crt-client`

#### Note

Tambahkan `<exclusions>` elemen ke semua dependensi klien layanan di file `Andapom.xml`.

## Konfigurasi klien layanan untuk memintas pencarian

Tentukan suatu wilayah

Saat Anda membuat klien layanan, panggil `region` metode pada pembuat klien layanan. Ini memintas [proses pencarian Wilayah](#) default SDK yang memeriksa beberapa tempat untuk informasi. Wilayah AWS

Untuk menjaga kode Lambda independen dari wilayah, gunakan kode berikut di dalam metode. `region` Kode ini mengakses variabel `AWS_REGION` lingkungan yang ditetapkan oleh wadah Lambda.

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

## Gunakan `EnvironmentVariableCredentialProvider`

Sama seperti perilaku pencarian default untuk informasi Wilayah, SDK mencari kredensial di beberapa tempat. Dengan menentukan `EnvironmentVariableCredentialProvider` kapan Anda membangun klien layanan, Anda menghemat waktu dalam proses pencarian SDK.

### Note

Menggunakan penyedia kredensi ini memungkinkan kode untuk digunakan dalam Lambda fungsi, tetapi mungkin tidak berfungsi pada Amazon EC2 atau sistem lain.

Cuplikan kode berikut menunjukkan klien layanan S3 yang dikonfigurasi dengan tepat untuk digunakan di lingkungan Lambda.

```
S3Client client = S3Client.builder()

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

## Inisialisasi klien SDK di luar penanganan fungsi Lambda

Sebaiknya inisialisasi klien SDK di luar metode penanganan Lambda. Dengan cara ini, jika konteks eksekusi digunakan kembali, inisialisasi klien layanan dapat dilewati. Dengan menggunakan kembali instance klien dan koneksinya, pemanggilan berikutnya dari metode handler terjadi lebih cepat.

Dalam contoh berikut, `S3Client` instance diinisialisasi dalam konstruktor menggunakan metode pabrik statis. Jika wadah yang dikelola oleh lingkungan Lambda digunakan kembali, instance yang diinisialisasi `S3Client` akan digunakan kembali.

```
public class App implements RequestHandler<Object, Object> {
    private final S3Client s3Client;

    public App() {
        s3Client = DependencyFactory.s3Client();
    }

    @Override
```

```
public Object handle Request(final Object input, final Context context) {
    ListBucketResponse response = s3Client.listBuckets();
    // Process the response.
}
}
```

## Minimalkan injeksi ketergantungan

Kerangka kerja injeksi ketergantungan (DI) mungkin membutuhkan waktu tambahan untuk menyelesaikan proses penyiapan. Mereka mungkin juga memerlukan dependensi tambahan, yang membutuhkan waktu untuk memuat.

Jika kerangka kerja DI diperlukan, kami sarankan menggunakan kerangka kerja DI ringan seperti [Dagger](#).

## Gunakan penargetan Maven Archetype AWS Lambda

Tim AWS Java SDK telah mengembangkan [template Maven Archetype](#) untuk mem-bootstrap proyek Lambda dengan waktu startup minimal. Anda dapat membangun proyek Maven dari pola dasar dan mengetahui bahwa dependensi dikonfigurasi sesuai untuk lingkungan Lambda.

[Untuk mempelajari lebih lanjut tentang arketipe dan bekerja melalui contoh penyebaran, lihat posting blog ini.](#)

## Pertimbangkan Lambda SnapStart untuk Java

Jika persyaratan runtime Anda kompatibel, AWS tawarkan [SnapStart Lambda](#) untuk Java. Lambda SnapStart adalah solusi berbasis infrastruktur yang meningkatkan kinerja startup untuk fungsi Java. Saat Anda mempublikasikan versi baru suatu fungsi, Lambda SnapStart menginisialisasinya dan mengambil snapshot memori dan status disk yang tidak dapat diubah dan terenkripsi. SnapStart kemudian menyimpan snapshot untuk digunakan kembali.

## Perubahan versi 2.x yang memengaruhi waktu startup

Selain perubahan yang Anda buat pada kode Anda, SDK for Java versi 2.x mencakup tiga perubahan utama yang mengurangi waktu startup:

- Penggunaan [jackson-jr](#), yang merupakan pustaka serialisasi yang meningkatkan waktu inisialisasi
- Penggunaan pustaka [java.time](#) untuk objek tanggal dan waktu, yang merupakan bagian dari JDK

- Penggunaan [SLF4j untuk fasad penebangan](#)

## Sumber daya tambahan

Panduan AWS Lambda Pengembang berisi [bagian tentang praktik terbaik](#) untuk mengembangkan fungsi Lambda yang tidak spesifik untuk Java.

Untuk contoh membangun aplikasi cloud-native di Java yang menggunakan AWS Lambda, lihat konten [lokakarya](#) ini. Optimalisasi kinerja diskusi lokakarya dan praktik terbaik lainnya.

Anda dapat mempertimbangkan untuk menggunakan gambar statis yang dikompilasi sebelumnya untuk mengurangi latensi startup. Misalnya, Anda dapat menggunakan SDK for Java 2.x dan Maven untuk [membuat](#) gambar asli GraalVM.

## Klien HTTP

Anda dapat mengubah klien HTTP untuk digunakan untuk klien layanan Anda serta mengubah konfigurasi default untuk klien HTTP dengan AWS SDK for Java 2.x. Bagian ini membahas klien HTTP dan pengaturan untuk SDK.

## Klien HTTP tersedia di SDK for Java

### Klien sinkron

Klien HTTP sinkron di SDK for Java mengimplementasikan [SdkHttpClient](#) antarmuka. Klien layanan sinkron, seperti `S3Client` atau `DynamoDbClient`, memerlukan penggunaan klien HTTP sinkron. Ini AWS SDK for Java menawarkan tiga klien HTTP sinkron.

#### ApacheHttpClient (default)

[ApacheHttpClient](#) adalah klien HTTP default untuk klien layanan sinkron. Untuk informasi tentang mengonfigurasi `ApacheHttpClient`, lihat [Konfigurasi klien HTTP berbasis Apache](#).

#### AwsCrhttpClient

[AwsCrhttpClient](#) menyediakan throughput tinggi dan IO non-pemblokiran. Ini dibangun di atas AWS Common Runtime (CRT) Http Client. Untuk informasi tentang mengkonfigurasi `AwsCrhttpClient` dan menggunakannya dengan klien layanan, lihat [the section called "Konfigurasi AWS klien HTTP berbasis CRT"](#).

## URLConnectionHttpClient

Untuk meminimalkan jumlah toples dan pustaka pihak ketiga yang Anda gunakan aplikasi, Anda dapat menggunakan [URLConnectionHttpClient](#). Untuk informasi tentang mengonfigurasi `URLConnectionHttpClient`, lihat [Konfigurasi klien HTTP berbasis URLConnection](#).

## Klien asinkron

Klien HTTP asinkron di SDK for Java mengimplementasikan antarmuka [SdkAsyncHttpClient](#). Klien layanan asinkron, seperti `S3AsyncClient` atau `DynamoDbAsyncClient`, memerlukan penggunaan klien HTTP asinkron. Ini AWS SDK for Java menawarkan dua klien HTTP asinkron.

### NettyNioAsyncHttpClient (default)

[NettyNioAsyncHttpClient](#) adalah klien HTTP default yang digunakan oleh klien asinkron. Untuk informasi tentang mengonfigurasi `NettyNioAsyncHttpClient`, lihat [the section called "Konfigurasi klien HTTP berbasis Netty"](#).

### AwsCrtAsyncHttpClient

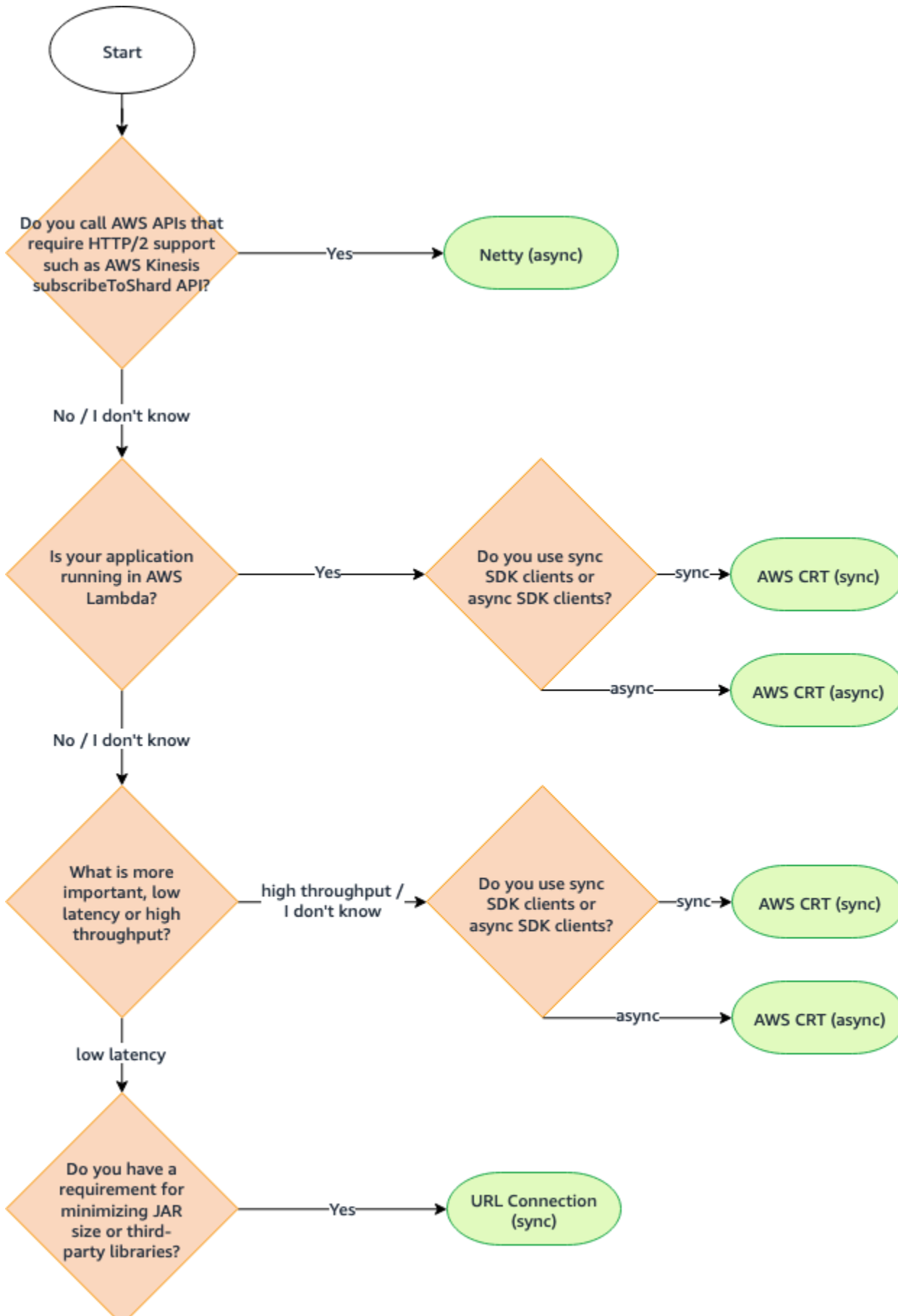
[AwsCrtAsyncHttpClient](#) ini didasarkan pada AWS Common Runtime (CRT) HTTP Client. Untuk informasi tentang mengonfigurasi `AwsCrtAsyncHttpClient`, lihat [the section called "Konfigurasi AWS klien HTTP berbasis CRT"](#).

## Rekomendasi klien HTTP

Beberapa faktor ikut bermain ketika Anda memilih implementasi klien HTTP. Gunakan informasi berikut untuk membantu Anda memutuskan.

### Diagram alur rekomendasi

Diagram alur berikut memberikan panduan umum untuk membantu Anda menentukan klien HTTP mana yang akan digunakan.



## Perbandingan klien HTTP

Tabel berikut memberikan informasi rinci untuk setiap klien HTTP.

Klien HTTP	Sinkronisasi atau asinkron	Kapan harus menggunakan	Batasan/k kelemahan
Klien HTTP berbasis Apache  (klien HTTP sinkronisasi default)	Sinkronkan	Gunakan jika Anda lebih suka latensi rendah daripada throughput tinggi	Waktu startup lebih lambat dibandingkan dengan klien HTTP lainnya
Klien HTTP berbasis URLConnection	Sinkronkan	Gunakan jika Anda memiliki persyaratan sulit untuk membatasi dependensi pihak ketiga	Tidak mendukung metode PATCH HTTP, yang diperlukan oleh beberapa API seperti operasi Pembaruan Amazon ApiGateway
AWS <sup>Klien HTTP</sup> sinkronisasi berbasis CRT 1	Sinkronkan	<ul style="list-style-type: none"> <li>Gunakan jika aplikasi Anda berjalan di AWS Lambda</li> <li>Gunakan jika Anda lebih suka throughput tinggi daripada latensi rendah</li> <li>Gunakan jika Anda lebih suka menyinkronkan klien SDK</li> </ul>	T/A
Klien HTTP berbasis Netty	Asinkron	<ul style="list-style-type: none"> <li>Gunakan jika aplikasi Anda memanggil API yang memerlukan dukungan HTTP/2 seperti Kinesis API <a href="#">SubscribeToShard</a></li> </ul>	Waktu startup lebih lambat dibandingkan

Klien HTTP	Sinkronisasi atau asinkron	Kapan harus menggunakan	Batasan/klemahan
(klien HTTP asinkron default)			dengan klien HTTP lainnya
AWS <sup>1</sup> Klien HTTP asinkron berbasis CRT	Asinkron	<ul style="list-style-type: none"> <li>Gunakan jika aplikasi Anda berjalan di AWS Lambda</li> <li>Gunakan jika Anda lebih suka throughput tinggi daripada latensi rendah</li> <li>Gunakan jika Anda lebih suka klien SDK async</li> </ul>	<ul style="list-style-type: none"> <li>Tidak mendukung klien layanan yang memerlukan dukungan HTTP/2 seperti dan KinesisAsyncClient TranscribeStreamingAsyncClient</li> </ul>

<sup>1</sup> Karena manfaat tambahannya, kami sarankan Anda menggunakan klien HTTP AWS berbasis CRT jika memungkinkan.

## Default konfigurasi cerdas

AWS SDK for Java 2.x (versi 2.17.102 atau yang lebih baru) menawarkan fitur default konfigurasi cerdas. Fitur ini mengoptimalkan dua properti klien HTTP bersama dengan properti lain yang tidak mempengaruhi klien HTTP.

Default konfigurasi cerdas menetapkan nilai yang masuk akal untuk `tlsNegotiationTimeoutInMillis` properti `connectTimeoutInMillis` dan berdasarkan nilai mode default yang Anda berikan. Anda memilih nilai mode default berdasarkan karakteristik aplikasi Anda.

[Untuk informasi selengkapnya tentang default konfigurasi cerdas dan cara memilih nilai mode default yang paling cocok untuk aplikasi Anda, lihat Panduan Referensi SDK dan Alat. AWS](#)

Berikut adalah empat cara untuk mengatur mode default untuk aplikasi Anda.



## Service client

Gunakan pembuat klien layanan untuk mengonfigurasi mode default langsung pada klien layanan. Contoh berikut menetapkan modus default untuk `auto`. `DynamoDbClient`

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

## System property

Anda dapat menggunakan properti `aws.defaultsMode` sistem untuk menentukan mode default. Jika Anda mengatur properti sistem di Java, Anda perlu mengatur properti sebelum menginisialisasi klien layanan apa pun.

Contoh berikut menunjukkan kepada Anda cara mengatur mode default untuk `auto` menggunakan properti sistem yang disetel di Java.

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

Contoh berikut menunjukkan bagaimana Anda mengatur mode default untuk `auto` menggunakan `-D` opsi perintah. `java`

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

## Environment variable

Tetapkan nilai untuk variabel lingkungan `AWS_DEFAULTS_MODE` untuk memilih mode default untuk aplikasi Anda.

Informasi berikut menunjukkan perintah untuk menjalankan untuk mengatur nilai untuk modus default untuk `auto` menggunakan variabel lingkungan.

Sistem operasi	Perintah untuk mengatur variabel lingkungan
Linux, macOS, atau Unix	<code>export AWS_DEFAULTS_MODE=auto</code>
Windows	<code>set AWS_DEFAULTS_MODE=auto</code>

## AWS config file

Anda dapat menambahkan properti `defaults_mode` konfigurasi ke AWS config file bersama seperti yang ditunjukkan contoh berikut.

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

Jika Anda menyetel mode default secara global dengan properti sistem, variabel lingkungan, atau file AWS konfigurasi, Anda dapat mengganti pengaturan saat membuat klien HTTP.

Saat Anda membangun klien HTTP dengan `httpClientBuilder()` metode ini, pengaturan hanya berlaku untuk instance yang sedang Anda bangun. Contoh ini ditunjukkan [di sini](#). Klien HTTP berbasis Netty dalam contoh ini mengganti nilai mode default apa pun yang ditetapkan secara global untuk `connectTimeoutInMillis` `tlsNegotiationTimeoutInMillis`

## Dukungan proxy

Anda dapat mengkonfigurasi proxy HTTP dengan menggunakan kode, dengan mengatur properti sistem Java, atau dengan menggabungkan kedua pendekatan. SDK saat ini tidak mendukung variabel lingkungan untuk mengonfigurasi proxy.

Anda mengonfigurasi proxy dalam kode dengan `ProxyConfiguration` pembuat khusus klien saat Anda membangun klien layanan. Bagian untuk setiap klien HTTP dalam topik ini menunjukkan contoh konfigurasi proxy. [Contoh ini untuk klien HTTP Apache](#).

Dukungan klien HTTP dari properti sistem Java untuk proxy HTTP

Properti sistem	Deskripsi	Dukungan klien HTTP
<code>http.ProxyHost</code>	Nama host dari server proxy HTTP	Semua
<code>http.Proxyport</code>	Nomor port server proxy HTTP	Semua
<code>http.ProxyUser</code>	Nama pengguna untuk otentikasi proxy HTTP	Semua
<code>http.ProxyPassword</code>	Kata sandi untuk otentikasi proxy HTTP	Semua

Properti sistem	Deskripsi	Dukungan klien HTTP
<code>http.nonProxyHosts</code>	Daftar host yang harus dijangkau secara langsung, melewati proxy. <a href="#">Juga berlaku ketika HTTPS digunakan.</a>	Semua
<code>https.proxyhost</code>	Nama host dari server proxy HTTPS	Netty, CRT
<code>https.proxyport</code>	Nomor port server proxy HTTPS	Netty, CRT
<code>https.proxyuser</code>	Nama pengguna untuk otentikasi proxy HTTPS	Netty, CRT
<code>https.proxyPassword</code>	Kata sandi untuk otentikasi proxy HTTPS	Netty, CRT

Istilah yang digunakan dalam tabel berarti:

- Semua: Semua klien HTTP yang ditawarkan oleh SDK  
—`URLConnectionHttpClient`, `ApacheHttpClient`, `NettyNioAsyncHttpClient`  
`AwsCrtAsyncHttpClient`
- Netty: klien HTTP berbasis Netty () `NettyNioAsyncHttpClient`
- CRT: klien HTTP AWS berbasis CRT, (dan) `AwsCrtHttpClient` `AwsCrtAsyncHttpClient`

Anda dapat menggunakan campuran konfigurasi klien HTTP dan properti sistem. Setiap `ProxyConfiguration` pembangun klien HTTP menyediakan `useSystemPropertyValues` pengaturan. Secara default, pengaturannya adalah `true`. Saat pengaturannya `true`, SDK secara otomatis menggunakan nilai properti sistem untuk opsi yang tidak disediakan dengan menggunakan `ProxyConfiguration` pembuat.

Contoh berikut menunjukkan konfigurasi yang disediakan oleh properti sistem dan kode.

```
// Command line with the proxy password set as a system property.
$ java -Dhttp.proxyPassword=password -cp ... App
```

```
// Since the 'useSystemPropertyValues' setting is 'true' (the default), the SDK will
// supplement
// the proxy configuration in code with the 'http.proxyPassword' value from the system
// property.
SdkHttpClient apacheHttpClient = ApacheHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .endpoint(URI.create("http://localhost:1234"))
        .username("username")
        .build())
    .build();

// Use the apache HTTP client with proxy configuration.
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
    .httpClient(apacheHttpClient)
    .build();
```

### Note

Alih-alih mengatur endpoint properti dalam kode seperti yang ditunjukkan pada cuplikan kode sebelumnya, Anda dapat menggunakan properti sistem berikut.

```
-Dhttp.proxyHost=localhost -Dhttp.proxyPort=1234
```

## Konfigurasi klien HTTP berbasis Apache

Klien layanan sinkron dalam AWS SDK for Java 2.x menggunakan klien HTTP berbasis Apache, secara default. [ApacheHttpClient](#) SDK ApacheHttpClient didasarkan pada [HttpClient](#) Apache.

SDK juga menawarkan [URLConnectionHttpClient](#), yang memuat lebih cepat, tetapi memiliki lebih sedikit fitur. Untuk informasi tentang mengonfigurasi [URLConnectionHttpClient](#), lihat [the section called “Konfigurasi klien HTTP berbasis URLConnection”](#).

Untuk melihat set lengkap opsi konfigurasi yang tersedia untuk Anda [ApacheHttpClient](#), lihat [ApacheHttpClient.Builder](#) dan [ProxyConfiguration.Builder](#).

## Akses `ApacheHttpClient`

Dalam kebanyakan situasi, Anda menggunakan `ApacheHttpClient` tanpa konfigurasi eksplisit. Anda mendeklarasikan klien layanan Anda dan SDK akan mengonfigurasi `ApacheHttpClient` dengan nilai standar untuk Anda.

Jika Anda ingin mengkonfigurasi `ApacheHttpClient` atau menggunakannya secara eksplisit dengan beberapa klien layanan, Anda harus membuatnya tersedia untuk konfigurasi.

### Tidak diperlukan konfigurasi

Saat Anda mendeklarasikan dependensi pada klien layanan di Maven, SDK menambahkan dependensi runtime pada artefak. `apache-client` Ini membuat `ApacheHttpClient` kelas tersedia untuk kode Anda saat runtime, tetapi tidak pada waktu kompilasi. Jika Anda tidak mengonfigurasi klien HTTP berbasis Apache, Anda tidak perlu menentukan ketergantungan untuk itu.

Dalam cuplikan XHTML berikut dari `pom.xml` file Maven, dependensi yang dideklarasikan dengan `<artifactId>s3</artifactId>` secara otomatis membawa klien HTTP berbasis Apache. Anda tidak perlu mendeklarasikan ketergantungan khusus untuk itu.

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

Dengan dependensi ini, Anda tidak dapat membuat perubahan konfigurasi HTTP eksplisit, karena `ApacheHttpClient` pustaka hanya ada di classpath runtime.

### Konfigurasi diperlukan

Untuk mengkonfigurasi `ApacheHttpClient`, Anda perlu menambahkan ketergantungan pada `apache-client` perpustakaan pada waktu kompilasi.

Lihat contoh berikut dari file Maven untuk mengkonfigurasi pom.xml file. `ApacheHttpClient`

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

## Gunakan dan konfigurasi `ApacheHttpClient`

Anda dapat mengonfigurasi instance `ApacheHttpClient` bersama dengan membangun klien layanan, atau Anda dapat mengonfigurasi satu instance untuk dibagikan di beberapa klien layanan.

Dengan salah satu pendekatan, Anda menggunakan [ApacheHttpClient.Builder](#) untuk mengkonfigurasi properti untuk klien HTTP berbasis Apache.

Praktik terbaik: mendedikasikan sebuah `ApacheHttpClient` instance untuk klien layanan

Jika Anda perlu mengonfigurasi instance `ApacheHttpClient`, kami sarankan Anda membuat `ApacheHttpClient` instance khusus. Anda dapat melakukannya dengan menggunakan `httpClientBuilder` metode pembangun klien layanan. Dengan cara ini, siklus hidup klien HTTP dikelola oleh SDK, yang membantu menghindari potensi kebocoran memori jika `ApacheHttpClient` instance tidak ditutup saat tidak lagi diperlukan.

Contoh berikut membuat `S3Client` dan mengkonfigurasi instance tertanam `ApacheHttpClient` dengan `maxConnections` dan `connectionTimeout` nilai-nilai. Contoh HTTP dibuat menggunakan `httpClientBuilder` metode `S3Client.Builder`.

## Impor

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

## Kode

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

// Perform work with the s3Client.

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

## Pendekatan alternatif: bagikan `ApacheHttpClient` contoh

Untuk membantu menjaga penggunaan sumber daya dan memori lebih rendah untuk aplikasi Anda, Anda dapat mengonfigurasi `ApacheHttpClient` dan membagikannya di beberapa klien layanan. Kumpulan koneksi HTTP akan dibagikan, yang menurunkan penggunaan sumber daya.

### Note

Ketika sebuah `ApacheHttpClient` instance dibagikan, Anda harus menutupnya ketika sudah siap untuk dibuang. SDK tidak akan menutup instance saat klien layanan ditutup.

Contoh berikut mengkonfigurasi klien HTTP berbasis Apache yang digunakan oleh dua klien layanan. `ApacheHttpClientInstance` yang dikonfigurasi diteruskan ke `httpClient` metode masing-masing builder. Ketika klien layanan dan klien HTTP tidak lagi diperlukan, kode secara eksplisit menutupnya. Kode menutup klien HTTP terakhir.

## Impor

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

## Kode

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

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

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

// Perform work with the s3Client and dynamoDbClient.

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

## Contoh konfigurasi proxy

Cuplikan kode berikut menggunakan [pembangun konfigurasi proxy untuk klien HTTP Apache](#).

```
SdkHttpClient apacheHttpClient = ApacheHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .endpoint(URI.create("http://example.com:1234"))
        .username("username")
        .password("password")
        .addNonProxyHost("localhost")
        .addNonProxyHost("host.example.com")
        .build())
    .build();
```

Properti sistem Java yang setara untuk konfigurasi proxy ditampilkan dalam cuplikan baris perintah berikut.



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

### Note

Klien HTTP Apache saat ini tidak mendukung properti sistem proxy HTTPS.

## Konfigurasi klien HTTP berbasis URLConnection

Ini AWS SDK for Java 2.x menawarkan klien [URLConnectionHttpClient](#) HTTP yang lebih ringan dibandingkan dengan default. `ApacheHttpClient` `URLConnectionHttpClient` ini didasarkan pada Java [URLConnection](#).

`URLConnectionHttpClient` lebih cepat daripada klien HTTP berbasis Apache, tetapi memiliki lebih sedikit fitur. Karena memuat lebih cepat, ini adalah [solusi yang baik](#) untuk AWS Lambda fungsi Java.

Ini `URLConnectionHttpClient` memiliki beberapa [opsi yang dapat dikonfigurasi](#) yang dapat Anda akses.

### Note

`URLConnectionHttpClient` tidak mendukung metode HTTP PATCH. Beberapa operasi AWS API memerlukan permintaan PATCH. Nama-nama operasi itu biasanya dimulai dengan `Update*`. Berikut ini adalah beberapa contoh.

- [Beberapa Update\\* operasi](#) di AWS Security Hub API dan juga [BatchUpdateFindings](#) operasi
- Semua [Update\\* operasi](#) API Amazon API Gateway
- [Beberapa Update\\* operasi](#) di Amazon WorkDocs API

Jika Anda mungkin menggunakan `URLConnectionHttpClient`, pertama-tama lihat Referensi API untuk Layanan AWS yang Anda gunakan. Periksa untuk melihat apakah operasi yang Anda butuhkan menggunakan operasi PATCH.

## Akses `URLConnectionHttpClient`

Untuk mengkonfigurasi dan menggunakan `URLConnectionHttpClient`, Anda mendeklarasikan ketergantungan pada artefak `url-connection-client` Maven dalam file `Anda.pom.xml`

Berbeda dengan `ApacheHttpClient`, tidak `URLConnectionHttpClient` secara otomatis ditambahkan ke proyek Anda, jadi penggunaan harus secara khusus mendeklarasikannya.

Contoh berikut dari `pom.xml` file menunjukkan dependensi yang diperlukan untuk menggunakan dan mengkonfigurasi klien HTTP.

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

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

<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>url-connection-client</artifactId>
  </dependency>
</dependencies>
```

## Gunakan dan konfigurasi `URLConnectionHttpClient`

Anda dapat mengonfigurasi instance `URLConnectionHttpClient` bersama dengan membangun klien layanan, atau Anda dapat mengonfigurasi satu instance untuk dibagikan di beberapa klien layanan.

Dengan salah satu pendekatan, Anda menggunakan [URLConnectionHttpClient.Builder](#) untuk mengonfigurasi properti untuk klien HTTP berbasis `URLConnection`.

Praktik terbaik: mendedikasikan sebuah **URLConnectionHttpClient** instance untuk klien layanan

Jika Anda perlu mengonfigurasi instance `URLConnectionHttpClient`, kami sarankan Anda membuat `URLConnectionHttpClient` instance khusus. Anda dapat melakukannya dengan menggunakan `httpClientBuilder` metode pembangun klien layanan. Dengan cara ini, siklus hidup klien HTTP dikelola oleh SDK, yang membantu menghindari potensi kebocoran memori jika `URLConnectionHttpClient` instance tidak ditutup saat tidak lagi diperlukan.

Contoh berikut membuat `S3Client` dan mengkonfigurasi instance tertanam `URLConnectionHttpClient` dengan `socketTimeout` dan `proxyConfiguration` nilai-nilai. `proxyConfiguration` metode ini mengambil ekspresi Java lambda tipe `Consumer<ProxyConfiguration.Builder>`.

Impor

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

Kode

```
// Singleton: Use the s3Client for all requests.
S3Client s3Client =
    S3Client.builder()
        .httpClientBuilder(URLConnectionHttpClient.builder()
            .socketTimeout(Duration.ofMinutes(5))
            .proxyConfiguration(proxy -> proxy.endpoint(URI.create("http://
proxy.mydomain.net:8888"))))
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();

// Perform work with the s3Client.

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

Pendekatan alternatif: bagikan **URLConnectionHttpClient** contoh

Untuk membantu menjaga penggunaan sumber daya dan memori lebih rendah untuk aplikasi Anda, Anda dapat mengonfigurasi `URLConnectionHttpClient` dan membagikannya di beberapa klien layanan. Kumpulan koneksi HTTP akan dibagikan, yang menurunkan penggunaan sumber daya.

**Note**

Ketika sebuah `URLConnectionHttpClient` instance dibagikan, Anda harus menutupnya ketika sudah siap untuk dibuang. SDK tidak akan menutup instance saat klien layanan ditutup.

Contoh berikut mengkonfigurasi klien HTTP berbasis `URLConnection` yang digunakan oleh dua klien layanan. `URLConnectionHttpClientInstance` yang dikonfigurasi diteruskan ke `httpClient` metode masing-masing builder. Ketika klien layanan dan klien HTTP tidak lagi diperlukan, kode secara eksplisit menutupnya. Kode menutup klien HTTP terakhir.

**Impor**

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

**Kode**

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

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

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

```
        .build();

// Perform work with the s3Client and dynamoDbClient.

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

## Gunakan **URLConnectionHttpClient** dan **ApacheHttpClient** bersama-sama

Ketika Anda menggunakan `URLConnectionHttpClient` dalam aplikasi Anda, Anda harus menyediakan setiap klien layanan dengan `URLConnectionHttpClient` instance atau `ApacheHttpClient` instance menggunakan `httpClientBuilder` metode pembuat klien layanan.

Pengecualian terjadi jika program Anda menggunakan beberapa klien layanan dan kedua hal berikut ini benar:

- Satu klien layanan dikonfigurasi untuk menggunakan `URLConnectionHttpClient` instance
- Klien layanan lain menggunakan default `ApacheHttpClient` tanpa secara eksplisit membangunnya dengan metode `or httpClient() httpClientBuilder()`

Pengecualian akan menyatakan bahwa beberapa implementasi HTTP ditemukan di classpath.

Contoh cuplikan kode berikut mengarah ke pengecualian.

```
// The dynamoDbClient uses the UrlConnectionHttpClient
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
    .httpClient(URLConnectionHttpClient.create())
    .build();

// The s3Client below uses the ApacheHttpClient at runtime, without specifying it.
// An SdkClientException is thrown with the message that multiple HTTP implementations
// were found on the classpath.
S3Client s3Client = S3Client.create();

// Perform work with the s3Client and dynamoDbClient.

dynamoDbClient.close();
s3Client.close();
```

## Hindari pengecualian dengan secara eksplisit mengonfigurasi dengan file. S3Client ApacheHttpClient

```
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
    .httpClient(URLConnectionHttpClient.create())
    .build();

S3Client s3Client = S3Client.builder()
    .httpClient(ApacheHttpClient.create()) // Explicitly build the
    ApacheHttpClient.
    .build();

// Perform work with the s3Client and dynamoDbClient.

dynamoDbClient.close();
s3Client.close();
```

### Note

Untuk secara eksplisit membuat `ApacheHttpClient`, Anda harus [menambahkan ketergantungan](#) pada `apache-client` artefak dalam file proyek Maven Anda.

## Contoh konfigurasi proxy

Cuplikan kode berikut menggunakan [pembangun konfigurasi proxy untuk klien HTTP koneksi URL](#).

```
SdkHttpClient urlHttpClient = UrlConnectionHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .endpoint(URI.create("http://example.com:1234"))
        .username("username")
        .password("password")
        .addNonProxyHost("localhost")
        .addNonProxyHost("host.example.com")
        .build())
    .build();
```

Properti sistem Java yang setara untuk konfigurasi proxy ditampilkan dalam cuplikan baris perintah berikut.

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

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

### Note

Koneksi URL klien HTTP saat ini tidak mendukung properti sistem proxy HTTPS.

## Konfigurasi klien HTTP berbasis Netty

Klien HTTP default untuk operasi asinkron dalam AWS SDK for Java 2.x adalah berbasis Netty. [NettyNioAsyncHttpClient](#) [Klien berbasis Netty didasarkan pada kerangka kerja jaringan berbasis peristiwa asinkron dari proyek Netty.](#)

Sebagai klien HTTP alternatif, Anda dapat menggunakan klien [HTTP AWS berbasis CRT](#) baru. Topik ini menunjukkan kepada Anda cara mengkonfigurasi `fileNettyNioAsyncHttpClient`.

### Akses `NettyNioAsyncHttpClient`

Dalam kebanyakan situasi, Anda menggunakan `NettyNioAsyncHttpClient` tanpa konfigurasi eksplisit dalam program asinkron. Anda mendeklarasikan klien layanan asinkron Anda dan SDK akan mengkonfigurasi dengan nilai standar untuk `NettyNioAsyncHttpClient` Anda.

Jika Anda ingin mengkonfigurasi `NettyNioAsyncHttpClient` atau menggunakannya secara eksplisit dengan beberapa klien layanan, Anda harus membuatnya tersedia untuk konfigurasi.

Tidak diperlukan konfigurasi

Saat Anda mendeklarasikan dependensi pada klien layanan di Maven, SDK menambahkan dependensi runtime pada artefak. `netty-nio-client` Ini membuat `NettyNioAsyncHttpClient` kelas tersedia untuk kode Anda saat runtime, tetapi tidak pada waktu kompilasi. Jika Anda tidak mengonfigurasi klien HTTP berbasis Netty, Anda tidak perlu menentukan ketergantungan untuk itu.

Dalam cuplikan XHTML berikut dari `pom.xml` file Maven, dependensi yang dideklarasikan dengan `<artifactId>dynamodb-enhanced</artifactId>` transitif membawa klien HTTP berbasis Netty. Anda tidak perlu mendeklarasikan dependensi khusus untuk itu.

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

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

Dengan dependensi ini, Anda tidak dapat membuat perubahan konfigurasi HTTP apa pun, karena `NettyNioAsyncHttpClient` pustaka hanya ada di classpath runtime.

### Konfigurasi diperlukan

Untuk mengkonfigurasi `NettyNioAsyncHttpClient`, Anda perlu menambahkan ketergantungan pada `netty-nio-client` artefak pada waktu kompilasi.

Lihat contoh berikut dari file Maven untuk mengkonfigurasi `pom.xml` file.

### `NettyNioAsyncHttpClient`

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



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

## Gunakan dan konfigurasi **NettyNioAsyncHttpClient**

Anda dapat mengonfigurasi instance `NettyNioAsyncHttpClient` bersama dengan membangun klien layanan, atau Anda dapat mengonfigurasi satu instance untuk dibagikan di beberapa klien layanan.

Dengan salah satu pendekatan, Anda menggunakan [NettyNioAsyncHttpClient.Builder](#) untuk mengonfigurasi properti untuk instance klien HTTP berbasis Netty.

Praktik terbaik: mendedikasikan **NettyNioAsyncHttpClient** instance untuk klien layanan

Jika Anda perlu mengonfigurasi instance `NettyNioAsyncHttpClient`, sebaiknya Anda membuat `NettyNioAsyncHttpClient` instance khusus. Anda dapat melakukannya dengan menggunakan `httpClientBuilder` metode pembangun klien layanan. Dengan cara ini, siklus hidup klien HTTP dikelola oleh SDK, yang membantu menghindari potensi kebocoran memori jika `NettyNioAsyncHttpClient` instance tidak ditutup saat tidak lagi diperlukan.

Contoh berikut menciptakan sebuah `DynamoDbAsyncClient` instance yang digunakan oleh sebuah `DynamoDbEnhancedAsyncClient` instance. `DynamoDbAsyncClient` instance berisi `NettyNioAsyncHttpClient` instance dengan `connectionTimeout` dan `maxConcurrency` nilai-nilai. Contoh HTTP dibuat menggunakan `httpClientBuilder` metode `DynamoDbAsyncClient.Builder`.

Impor

```
import software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.awscore.defaults.DefaultsMode;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedAsyncClient;
import
    software.amazon.awssdk.enhanced.dynamodb.extensions.AutoGeneratedTimestampRecordExtension;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.time.Duration;
```

## Kode

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

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

// Perform work with the dynamoDbAsyncClient and enhancedClient.

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

### Pendekatan alternatif: bagikan **NettyNioAsyncHttpClient** contoh

Untuk membantu menjaga penggunaan sumber daya dan memori lebih rendah untuk aplikasi Anda, Anda dapat mengonfigurasi `NettyNioAsyncHttpClient` dan membagikannya di beberapa klien layanan. Kumpulan koneksi HTTP akan dibagikan, yang menurunkan penggunaan sumber daya.

#### Note

Ketika sebuah `NettyNioAsyncHttpClient` instance dibagikan, Anda harus menutupnya ketika sudah siap untuk dibuang. SDK tidak akan menutup instance saat klien layanan ditutup.

Contoh berikut mengkonfigurasi klien HTTP berbasis Netty yang digunakan oleh dua klien layanan. `NettyNioAsyncHttpClientInstance` yang dikonfigurasi diteruskan ke `httpClient` metode

masing-masing builder. Ketika klien layanan dan klien HTTP tidak lagi diperlukan, kode secara eksplisit menutupnya. Kode menutup klien HTTP terakhir.

## Impor

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

## Kode

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

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

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

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

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

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

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

// Requests completed: Close all service clients.
s3AsyncClient.close();
dbAsyncClient.close();
```

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

## Contoh konfigurasi proxy

Cuplikan kode berikut menggunakan [pembangun konfigurasi proxy untuk klien Netty HTTP](#).

```
SdkAsyncHttpClient nettyHttpClient = NettyNioAsyncHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .host("myproxy")
        .port(1234)
        .username("username")
        .password("password")
        .build())
    .build();
```

Properti sistem Java yang setara untuk konfigurasi proxy ditampilkan dalam cuplikan baris perintah berikut.

```
$ java -Dhttps.proxyHost=myproxy -Dhttps.proxyPort=1234 -Dhttps.proxyUser=username \
-Dhttps.proxyPassword=password -cp ... App
```

### Important

Untuk menggunakan salah satu properti sistem proxy HTTPS, `scheme` properti harus disetel dalam kode `https`. Jika properti skema tidak disetel dalam kode, skema default ke HTTP dan SDK hanya mencari properti sistem. `http.*`

## Konfigurasi AWS klien HTTP berbasis CRT

Klien HTTP AWS berbasis CRT termasuk sinkron dan asinkron.

[AwsCrtHttpClient](#) dan [AwsCrtAsyncHttpClient](#) Klien HTTP AWS berbasis CRT memberikan manfaat klien HTTP berikut:

- Waktu startup SDK lebih cepat
- Jejak memori yang lebih kecil
- Mengurangi waktu latensi
- Manajemen kesehatan koneksi

- Penyeimbangan beban DNS

## AWSKomponen berbasis CRT di SDK

Klien HTTP AWS berbasis CRT, dijelaskan dalam topik ini, dan klien S3 AWS berbasis CRT adalah komponen yang berbeda dalam SDK.

Klien HTTP AWS berbasis CRT sinkron dan asinkron adalah implementasi antarmuka klien HTTP SDK dan digunakan untuk komunikasi HTTP umum. Mereka adalah alternatif untuk klien HTTP sinkron atau asinkron lainnya di SDK dengan manfaat tambahan.

[Klien S3 AWS berbasis CRT](#) adalah implementasi `AsyncClient` antarmuka [S3](#) dan digunakan untuk bekerja dengan layanan Amazon S3. Ini adalah alternatif untuk implementasi `S3AsyncClient` antarmuka berbasis Java dan menawarkan beberapa keuntungan.

Meskipun kedua komponen menggunakan pustaka dari [AWSCommon Runtime](#), klien HTTP AWS berbasis CRT tidak menggunakan [pustaka aws-c-s 3](#) dan tidak mendukung fitur API unggahan multipart [S3](#). Klien S3 AWS berbasis CRT, sebaliknya, dibuat khusus untuk mendukung fitur API unggahan multibagian S3.

## Akses klien HTTP AWS berbasis CRT

Sebelum Anda dapat menggunakan klien HTTP AWS berbasis CRT, tambahkan `aws-crt-client` artefak dengan versi minimum 2.22.0 ke dependensi proyek Anda.

Maven berikut `pom.xml` menunjukkan klien HTTP AWS berbasis CRT yang dideklarasikan menggunakan mekanisme bill of materials (BOM).

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

```
</dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>aws-crt-client</artifactId>
  </dependency>
</dependencies>
</project>
```

[Kunjungi repositori pusat Maven untuk versi terbaru.](#)

## Gunakan dan konfigurasi klien AWS HTTP berbasis CRT

Anda dapat mengonfigurasi klien HTTP AWS berbasis CRT bersama dengan membangun klien layanan, atau Anda dapat mengonfigurasi satu instance untuk dibagikan di beberapa klien layanan.

Dengan salah satu pendekatan, Anda menggunakan pembangun untuk [mengonfigurasi properti](#) untuk instance klien HTTP AWS berbasis CRT.

Praktik terbaik: mendedikasikan sebuah instance untuk klien layanan

Jika Anda perlu mengonfigurasi instance klien HTTP AWS berbasis CRT, kami sarankan Anda mendedikasikan instance dengan membangunnya bersama dengan klien layanan. Anda dapat melakukannya dengan menggunakan `httpClientBuilder` metode pembangun klien layanan. Dengan cara ini, siklus hidup klien HTTP dikelola oleh SDK, yang membantu menghindari potensi kebocoran memori jika instance klien HTTP AWS berbasis CRT tidak ditutup ketika tidak lagi diperlukan.

Contoh berikut membuat klien layanan S3 dan mengkonfigurasi klien HTTP AWS berbasis CRT dengan dan nilai-nilai. `connectionTimeout` `maxConcurrency`

### Synchronous client

#### Impor

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

#### Kode

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

// Perform work with the s3Client.

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

## Asynchronous client

### Impor

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

### Kode

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

// Perform work with the s3AsyncClient.

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

## Pendekatan alternatif: bagikan contoh

Untuk membantu menjaga penggunaan sumber daya dan memori lebih rendah untuk aplikasi Anda, Anda dapat mengonfigurasi klien HTTP AWS berbasis CRT dan membagikannya di beberapa klien layanan. Kumpulan koneksi HTTP akan dibagikan, yang menurunkan penggunaan sumber daya.

**Note**

Ketika instance klien HTTP AWS berbasis CRT dibagikan, Anda harus menutupnya ketika sudah siap untuk dibuang. SDK tidak akan menutup instance saat klien layanan ditutup.

Contoh berikut mengkonfigurasi instance klien HTTP AWS berbasis CRT dengan `connectionTimeout` dan nilai-nilai `maxConcurrency` Instance yang dikonfigurasi diteruskan ke `httpClient` metode setiap pembuat klien layanan. Ketika klien layanan dan klien HTTP tidak lagi diperlukan, mereka secara eksplisit ditutup. Klien HTTP ditutup terakhir.

**Synchronous client****Impor**

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

**Kode**

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

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

DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
```



```
.httpClient(crtHttpClient)
.credentialsProvider(EnvironmentVariableCredentialsProvider.crea
.defaultsMode(DefaultsMode.IN_REGION)
.region(Region.US_EAST_1)
.build();

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

## Asynchronous client

### Impor

```
import
software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.awscore.defaultsmode.DefaultsMode;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.crt.AwsCrtAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import java.time.Duration;
```

### Kode

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

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

DynamoDbAsyncClient dynamoDbAsyncClient = DynamoDbAsyncClient.builder()
.httpClient(crtAsyncHttpClient)
```

```

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
.defaultsMode(DefaultsMode.IN_REGION)
.region(Region.US_EAST_1)
.build();

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

```

## Tetapkan klien HTTP AWS berbasis CRT sebagai default

Anda dapat mengatur file build Maven agar SDK menggunakan klien HTTP AWS berbasis CRT sebagai klien HTTP default untuk klien layanan.

Anda melakukan ini dengan menambahkan `exclusions` elemen dengan dependensi klien HTTP default ke setiap artefak klien layanan.

Dalam `pom.xml` contoh berikut, SDK menggunakan klien HTTP AWS berbasis CRT untuk layanan S3. Jika klien layanan dalam kode Anda adalah `S3AsyncClient`, SDK akan menggunakan `AwsCrtAsyncHttpClient`. Jika klien layanan adalah `S3Client`, SDK menggunakan `AwsCrtHttpClient`. Dengan pengaturan ini, klien HTTP asinkron berbasis Netty default dan HTTP sinkron berbasis Apache default tidak tersedia.

```

<project>
  <properties>
    <aws.sdk.version>VERSION</aws.sdk.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>s3</artifactId>
      <version>${aws.sdk.version}</version>
      <exclusions>
        <exclusion>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>netty-nio-client</artifactId>
        </exclusion>
        <exclusion>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>apache-client</artifactId>
        </exclusion>
      </exclusions>
    </dependency>
  </dependencies>
</project>

```

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

[Kunjungi repositori pusat Maven untuk nilai VERSION terbaru.](#)

#### Note

Jika beberapa klien layanan dideklarasikan dalam sebuah `pom.xml` file, semua memerlukan elemen `exclusions` XHTML.

## Menggunakan properti sistem Java

Untuk menggunakan klien HTTP AWS berbasis CRT sebagai HTTP default untuk aplikasi Anda, Anda dapat mengatur properti sistem Java `software.amazon.awssdk.http.async.service.impl` ke nilai `software.amazon.awssdk.http.crt.AwsCrtSdkHttpClient`.

Untuk mengatur selama startup aplikasi, jalankan perintah yang mirip dengan berikut ini.

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

Gunakan cuplikan kode berikut untuk mengatur properti sistem dalam kode aplikasi Anda.

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

#### Note

Anda perlu menambahkan ketergantungan pada `aws-crt-client` artefak dalam `pom.xml` file Anda ketika Anda menggunakan properti sistem untuk mengkonfigurasi penggunaan klien HTTP berbasis AWS CRT.

## Konfigurasi lanjutan dari klien AWS HTTP berbasis CRT

Anda dapat menggunakan berbagai pengaturan konfigurasi klien HTTP AWS berbasis CRT, termasuk konfigurasi kesehatan koneksi dan waktu idle maksimum. Anda dapat meninjau [opsi konfigurasi yang tersedia](#) untuk `AwsCrtAsyncHttpClient`. Anda dapat mengonfigurasi opsi yang sama untuk `fileAwsCrtHttpClient`.

### Konfigurasi kesehatan koneksi

Anda dapat mengonfigurasi konfigurasi kesehatan koneksi untuk klien HTTP AWS berbasis CRT dengan menggunakan `connectionHealthConfiguration` metode pada pembuat klien HTTP.

Contoh berikut membuat klien layanan S3 yang menggunakan instance klien HTTP AWS berbasis CRT yang dikonfigurasi dengan konfigurasi kesehatan koneksi dan waktu idle maksimum untuk koneksi.

### Synchronous client

#### Impor

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

#### Kode

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

// Perform work with s3Client.

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

## Asynchronous client

### Impor

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

### Kode

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

// Perform work with s3AsyncClient.

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

## Dukungan HTTP/2

Protokol HTTP/2 belum didukung di klien HTTP AWS berbasis CRT, tetapi direncanakan untuk rilis future.

Sementara itu, jika Anda menggunakan klien layanan yang memerlukan dukungan HTTP/2 seperti [KinesisAsyncClient](#) atau [TranscribeStreamingAsyncClient](#), pertimbangkan untuk menggunakan sebagai gantinya. [NettyNioAsyncHttpClient](#)

## Contoh konfigurasi proxy

Cuplikan kode berikut menunjukkan penggunaan [ProxyConfiguration.Builder](#) yang Anda gunakan untuk mengkonfigurasi pengaturan proxy dalam kode.

## Synchronous client

### Impor

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

## Kode

```
SdkHttpClient crtHttpClient = AwsCrtHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .host("myproxy")
        .port(1234)
        .username("username")
        .password("password")
        .build())
    .build();
```

## Asynchronous client

### Impor

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

## Kode

```
SdkAsyncHttpClient crtAsyncHttpClient = AwsCrtAsyncHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .host("myproxy")
        .port(1234)
        .username("username")
        .password("password")
        .build())
    .build();
```

Properti sistem Java yang setara untuk konfigurasi proxy ditampilkan dalam cuplikan baris perintah berikut.

```
$ java -Dhttps.proxyHost=myproxy -Dhttps.proxyPort=1234 -Dhttps.proxyUser=username \
```

```
-Dhttps.proxyPassword=password -cp ... App
```

### Important

Untuk menggunakan salah satu properti sistem proxy HTTPS, `scheme` properti harus disetel dalam kode `https`. Jika properti skema tidak disetel dalam kode, skema default ke HTTP dan SDK hanya mencari properti sistem. `http.*`

## Penanganan pengecualian untuk AWS SDK for Java 2.x

Memahami bagaimana dan kapan pengecualian AWS SDK for Java 2.x melempar penting untuk membangun aplikasi berkualitas tinggi menggunakan SDK. Bagian berikut menjelaskan berbagai kasus pengecualian yang dilemparkan oleh SDK dan cara menanganinya dengan tepat.

### Mengapa pengecualian yang tidak dicentang?

Pengecualian AWS SDK for Java menggunakan runtime (atau tidak dicentang) alih-alih pengecualian yang dicentang karena alasan berikut:

- Untuk memungkinkan pengembang mengontrol kesalahan yang ingin mereka tangani tanpa memaksa mereka untuk menangani kasus luar biasa yang tidak mereka khawatirkan (dan membuat kode mereka terlalu bertele-tele)
- Untuk mencegah masalah skalabilitas yang melekat pada pengecualian yang diperiksa dalam aplikasi besar

Secara umum, pengecualian yang diperiksa bekerja dengan baik pada skala kecil, tetapi dapat menjadi merepotkan karena aplikasi tumbuh dan menjadi lebih kompleks.

### `AwsServiceException` (dan subclass)

[AwsServiceException](#) adalah pengecualian paling umum yang akan Anda alami saat menggunakan AWS SDK for Java. `AwsServiceException` adalah subkelas yang lebih umum [SdkServiceException](#). `AwsServiceExceptions` mewakili respons kesalahan dari sebuah Layanan AWS. Misalnya, jika Anda mencoba menghentikan Amazon EC2 instance yang tidak ada, Amazon EC2 akan mengembalikan respons kesalahan dan semua detail respons kesalahan itu akan disertakan dalam `AwsServiceException` yang dilemparkan.

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

`AwsServiceException` memberi Anda informasi seperti:

- Kode status HTTP yang dikembalikan
- Kode AWS kesalahan yang dikembalikan
- Pesan kesalahan terperinci dari layanan di [AwsErrorDetails](#) kelas
- AWSID permintaan untuk permintaan yang gagal

Dalam kebanyakan kasus, subkelas khusus layanan dilemparkan untuk memungkinkan pengembang mengontrol `AwsServiceException` secara halus atas penanganan kasus kesalahan melalui blok catch. Referensi Java SDK API untuk [AwsServiceException](#) menampilkan sejumlah besar `AwsServiceException` subclass. Gunakan tautan subkelas untuk menelusuri untuk melihat pengecualian granular yang dilemparkan oleh layanan.

Misalnya, tautan berikut ke referensi SDK API menunjukkan hierarki pengecualian untuk beberapa hal umum. Layanan AWS Daftar subclass yang ditampilkan pada setiap halaman menunjukkan pengecualian spesifik yang dapat ditangkap oleh kode Anda.

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

Untuk mempelajari lebih lanjut tentang pengecualian, periksa `errorCode` pada [AwsErrorDetails](#) objek. Anda dapat menggunakan `errorCode` nilai untuk mencari informasi di API panduan layanan. Misalnya jika tertangkap dan `AwsErrorDetails#errorCode()` nilainya `InvalidRequest`, gunakan [daftar kode kesalahan](#) di Referensi API Amazon S3 untuk melihat detail selengkapnya. `S3Exception`

## SdkClientException

[SdkClientException](#) menunjukkan bahwa masalah terjadi di dalam kode klien Java, baik saat mencoba mengirim permintaan ke AWS atau saat mencoba mengurai respons dari AWS. An `SdkClientException` umumnya lebih parah daripada `SdkServiceException`, dan menunjukkan



masalah besar yang mencegah klien melakukan panggilan layanan ke AWS layanan. Misalnya, AWS SDK for Java melempar `SdkClientException` jika tidak ada koneksi jaringan yang tersedia ketika Anda mencoba memanggil operasi pada salah satu klien.

## Pengecualian dan perilaku coba lagi

SDK for Java mencoba ulang permintaan untuk [beberapa pengecualian sisi klien](#) dan [untuk kode status HTTP](#) yang diterimanya dari tanggapan. Layanan AWS Kesalahan ini ditangani sebagai bagian dari warisan `RetryMode` yang digunakan klien layanan secara default. Referensi Java API untuk [RetryMode](#) menjelaskan berbagai cara yang dapat Anda konfigurasi mode.

Untuk menyesuaikan pengecualian dan kode status HTTP yang memicu percobaan ulang otomatis, konfigurasi klien layanan Anda dengan [RetryPolicy](#) yang menambahkan [RetryOnExceptionsCondition](#) dan [RetryOnStatusCodeCondition](#) instance.

## Logging dengan SDK for Java 2.x

AWS SDK for Java 2.x menggunakan [SLF4J](#), yang merupakan lapisan abstraksi yang memungkinkan penggunaan salah satu dari beberapa sistem logging saat runtime.

Sistem logging yang didukung termasuk Java Logging Framework dan Apache [Log4j 2](#), antara lain. Topik ini menunjukkan cara menggunakan Log4j 2 sebagai sistem logging untuk bekerja dengan SDK.

## File konfigurasi Log4j 2

Anda biasanya menggunakan file konfigurasi, bernama `log4j2.xml` dengan Log4j 2. Contoh file konfigurasi ditunjukkan di bawah ini. Untuk mempelajari lebih lanjut tentang nilai yang digunakan dalam file konfigurasi, lihat [manual untuk konfigurasi Log4j](#).

`log4j2.xml` File harus berada di classpath saat aplikasi Anda dimulai. Untuk proyek Maven, letakkan file di direktori. `<project-dir>/src/main/resources`

File `log4j2.xml` konfigurasi menentukan properti seperti [tingkat logging](#), di mana output logging dikirim (misalnya, [ke file atau ke konsol](#)), dan [format output](#). Level logging menentukan tingkat detail yang keluaran Log4j 2. [Log4j 2 mendukung konsep beberapa hierarki logging](#). Level logging diatur secara independen untuk setiap hierarki. Hirarki logging utama yang Anda gunakan dengan AWS SDK for Java 2.x adalah `software.amazon.awssdk`.

## Tambahkan ketergantungan logging

Untuk mengonfigurasi pengikatan Log4j 2 untuk SLF4J di file build Anda, gunakan yang berikut ini.

### Maven

Tambahkan elemen berikut ke `pom.xml` file Anda.

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

### Gradle–Kotlin DSL

Tambahkan yang berikut ini ke `build.gradle.kts` file Anda.

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

Gunakan `2.20.0` untuk versi minimum `log4j-slf4j2-impl` artefak. Untuk versi terbaru, gunakan versi yang diterbitkan ke pusat [Maven](#). Ganti `VERSION` dengan versi yang akan Anda gunakan.

## Kesalahan dan peringatan khusus SDK

Sebaiknya Anda selalu membiarkan hierarki logger “`software.amazon.awssdk`” disetel ke “`WARN`” untuk menangkap pesan penting apa pun dari pustaka klien SDK. Misalnya, jika klien Amazon S3 mendeteksi bahwa aplikasi Anda belum menutup `InputStream` dan dapat membocorkan sumber daya dengan benar, klien S3 melaporkannya melalui pesan peringatan ke log. Ini juga memastikan bahwa pesan dicatat jika klien memiliki masalah dalam menangani permintaan atau tanggapan.

`log4j2.xmlFile` berikut menetapkan `rootLogger` ke “`WARN`”, yang menyebabkan peringatan dan pesan tingkat kesalahan dari semua logger dalam aplikasi menjadi output, termasuk yang ada

dalam hierarki “software.amazon.awssdk”. Atau, Anda dapat secara eksplisit mengatur hierarki logger “software.amazon.awssdk” ke “WARN” jika digunakan. `<Root level="ERROR">`

Contoh file konfigurasi Log4j2.xml

Konfigurasi ini akan mencatat pesan pada level “ERROR” dan “WARN” ke konsol untuk semua hierarki logger.

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

  <Loggers>
    <Root level="WARN">
      <AppenderRef ref="ConsoleAppender"/>
    </Root>
  </Loggers>
</Configuration>
```

## Pencatatan ringkasan permintaan/tanggapan

Setiap permintaan untuk Layanan AWS menghasilkan ID AWS permintaan unik yang berguna jika Anda mengalami masalah dengan Layanan AWS cara menangani permintaan. AWSID permintaan dapat diakses secara terprogram melalui [SdkServiceException](#) objek di SDK untuk panggilan layanan yang gagal, dan juga dapat dilaporkan melalui level log “DEBUG” dari logger “software.amazon.awssdk.request”.

log4j2.xml File berikut memungkinkan ringkasan permintaan dan tanggapan.

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

  <Loggers>
    <Root level="ERROR">
```

```

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

```

Berikut adalah contoh output log:

```

2022-09-23 16:02:08 [main] DEBUG software.amazon.awssdk.request:85 - Sending Request:
DefaultSdkHttpRequestFullRequest(httpMethod=POST, protocol=https, host=dynamodb.us-
east-1.amazonaws.com, encodedPath=/, headers=[amz-sdk-invocation-id, Content-Length,
Content-Type, User-Agent, X-Amz-Target], queryParameters=[])
2022-09-23 16:02:08 [main] DEBUG software.amazon.awssdk.request:85 - Received
successful response: 200, Request ID:
QS9DUMME2NHEDH8TGT9N5V530JVV4KQNS05AEMVJF66Q9ASUAAJG, Extended Request ID: not
available

```

Jika Anda tertarik hanya menggunakan ID permintaan `<Logger name="software.amazon.awssdk.requestId" level="DEBUG" />`.

## Penebangan kawat verbose

Hal ini dapat berguna untuk melihat permintaan dan tanggapan yang tepat bahwa SDK for Java 2.x mengirim dan menerima. Jika Anda memerlukan akses ke informasi ini, Anda dapat mengaktifkannya sementara dengan menambahkan konfigurasi yang diperlukan tergantung pada klien HTTP yang digunakan klien layanan.

Secara default, klien layanan sinkron, seperti [S3Client](#), menggunakan Apache yang mendasari [HttpClient](#), dan klien layanan asinkron, seperti [S3 AsyncClient](#), menggunakan klien HTTP non-blocking Netty.

Berikut adalah rincian klien HTTP yang dapat Anda gunakan untuk dua kategori klien layanan:

Klien HTTP sinkron	Klien HTTP Asinkron
<a href="#">ApacheHttpClient</a> (default)	<a href="#">NettyNioAsyncHttpClient</a> (default)
<a href="#">URLConnectionHttpClient</a>	<a href="#">AwsCrtAsyncHttpClient</a>

Lihat tab yang sesuai di bawah ini untuk pengaturan konfigurasi yang perlu Anda tambahkan tergantung pada klien HTTP yang mendasarinya.

### Warning

Kami menyarankan Anda hanya menggunakan wire logging untuk tujuan debugging. Nonaktifkan di lingkungan produksi Anda karena dapat mencatat data sensitif. Ini mencatat permintaan atau respons penuh tanpa enkripsi, bahkan untuk panggilan HTTPS. Untuk permintaan besar (misalnya, untuk mengunggah file ke Amazon S3) atau tanggapan, pencatatan kawat verbose juga dapat memengaruhi kinerja aplikasi Anda secara signifikan.

## ApacheHttpClient

Tambahkan logger “org.apache.http.wire” ke file `log4j2.xml` konfigurasi dan atur levelnya ke “DEBUG”.

`log4j2.xml`File berikut menyalakan pencatatan kawat penuh untuk Apache HttpClient.

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

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

Ketergantungan Maven tambahan pada `log4j-1.2-api` artefak diperlukan untuk wire logging dengan Apache karena menggunakan 1.2 di bawah kap.

Set lengkap dependensi Maven untuk `log4j 2`, termasuk wire logging untuk klien HTTP Apache ditampilkan dalam cuplikan file build berikut.

## Maven

```
...
<dependencyManagement>
  ...
  <dependencies>
    <dependency>
      <groupId>org.apache.logging.log4j</groupId>
      <artifactId>log4j-bom</artifactId>
      <version>VERSION</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
...
<!-- The following is needed for Log4j2 with SLF4J -->
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-slf4j2-impl</artifactId>
</dependency>

<!-- The following is needed for Apache HttpClient wire logging -->
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-1.2-api</artifactId>
</dependency>
...
```

## Gradle—Kotlin DSL

```
...
dependencies {
  ...
  implementation(platform("org.apache.logging.log4j:log4j-bom:VERSION"))
  implementation("org.apache.logging.log4j:log4j-slf4j2-impl")
  implementation("org.apache.logging.log4j:log4j-1.2-api")
}
...
```

Gunakan 2.20.0 untuk versi minimum log4j-bom artefak. Untuk versi terbaru, gunakan versi yang diterbitkan ke pusat [Maven](#). Ganti *VERSION* dengan versi yang akan Anda gunakan.

## URLConnectionHttpClient

Untuk mencatat detail untuk klien layanan yang menggunakan `URLConnectionHttpClient`, pertama buat `logging.properties` file dengan konten berikut:

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

Mengatur properti sistem JVM berikut dengan jalur lengkap dari: `logging.properties`

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

Konfigurasi ini akan mencatat satu-satunya header permintaan dan respons, misalnya:

```
<Request> FINE: sun.net.www.MessageHeader@35a9782c11 pairs: {GET /fileuploadtest
HTTP/1.1: null}{amz-sdk-invocation-id: 5f7e707e-4ac5-bef5-ba62-00d71034ffdc}
{amz-sdk-request: attempt=1; max=4}{Authorization: AWS4-HMAC-SHA256
Credential=<deleted>/20220927/us-east-1/s3/aws4_request, SignedHeaders=amz-sdk-
invocation-id;amz-sdk-request;host;x-amz-content-sha256;x-amz-date;x-amz-te,
Signature=e367fa0bc217a6a65675bb743e1280cf12f8e8d566196a816d948fdf0b42ca1a}{User-
Agent: aws-sdk-java/2.17.230 Mac_OS_X/12.5 OpenJDK_64-Bit_Server_VM/25.332-b08
Java/1.8.0_332 vendor/Amazon.com_Inc. io/sync http/URLConnection cfg/retry-mode/
legacy}{x-amz-content-sha256: UNSIGNED-PAYLOAD}{X-Amz-Date: 20220927T133955Z}{x-amz-
te: append-md5}{Host: tkhill-test1.s3.amazonaws.com}{Accept: text/html, image/gif,
image/jpeg, *; q=.2, */*; q=.2}{Connection: keep-alive}
<Response> FINE: sun.net.www.MessageHeader@70a36a6611 pairs: {null: HTTP/1.1
200 OK}{x-amz-id-2: sAFeZD0KdUMsBbkDjyDZw7P0oocb4C9KbiuzfJ6TWKQsGXHM/
dFu0vr2tUb7Y1wEHGdJ3DSIxq0=}{x-amz-request-id: P9QW9SMZ97FKZ9X7}{Date: Tue,
27 Sep 2022 13:39:57 GMT}{Last-Modified: Tue, 13 Sep 2022 14:38:12 GMT}{ETag:
"2cbe5ad4a064cedec33b452bebf48032"}{x-amz-transfer-encoding: append-md5}{Accept-
Ranges: bytes}{Content-Type: text/plain}{Server: AmazonS3}{Content-Length: 67}
```

Untuk melihat badan permintaan/respons, tambahkan `-Djavax.net.debug=all` ke properti JVM. Properti tambahan ini mencatat banyak informasi, termasuk semua informasi SSL.

Di dalam konsol log atau file log, cari "GET" atau "POST" dengan cepat pergi ke bagian log yang berisi permintaan dan tanggapan aktual. "Plaintext before ENCRYPTION" Cari permintaan dan tanggapan "Plaintext after DECRYPTION" untuk melihat teks lengkap header dan badan.

## NettyNioAsyncHttpClient

Jika klien layanan asinkron Anda menggunakan default `NettyNioAsyncHttpClient`, tambahkan dua logger tambahan ke `log4j2.xml` file Anda untuk mencatat header HTTP dan badan permintaan/respons.

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

Berikut adalah `log4j2.xml` contoh lengkapnya:

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

  <Loggers>
    <Root level="WARN">
      <AppenderRef ref="ConsoleAppender"/>
    </Root>
    <Logger name="software.amazon.awssdk" level="WARN" />
    <Logger name="software.amazon.awssdk.request" level="DEBUG" />
    <Logger name="io.netty.handler.logging" level="DEBUG" />
    <Logger name="io.netty.handler.codec.http2.Http2FrameLogger" level="DEBUG" /
  >
  </Loggers>
</Configuration>
```

Pengaturan ini mencatat semua detail header dan badan permintaan/respons.

## AwsCrtAsyncHttpClient

Jika Anda telah mengonfigurasi klien layanan Anda untuk menggunakan `instanceAwsCrtAsyncHttpClient`, Anda dapat mencatat detail dengan menyetel properti sistem JVM atau secara terprogram.

Log to a file at "Debug" level



### Menggunakan properti sistem:

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

### Secara terprogram:

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

// Execute this statement before constructing the
// SDK service client.
Log.initLoggingToFile(Log.LogLevel.Trace,
    "<path to file>");
```

Log to the console at "Debug" level

### Menggunakan properti sistem:

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

### Secara terprogram:

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

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

Untuk alasan keamanan, pada level "Trace" `AwsCrtAsyncHttpClient` log hanya header respons. Header permintaan, badan permintaan, dan badan respons tidak dicatat.

## Mengatur JVM TTL untuk pencarian nama DNS

Mesin virtual Java (JVM) menyimpan cache pencarian nama DNS. Ketika JVM menyelesaikan nama host ke alamat IP, itu menyimpan alamat IP untuk jangka waktu tertentu, yang dikenal sebagai (TTL). `time-to-live`

Karena AWS sumber daya menggunakan entri nama DNS yang terkadang berubah, kami sarankan Anda mengonfigurasi JVM Anda dengan nilai TTL tidak lebih dari 60 detik. Ini memastikan bahwa ketika alamat IP sumber daya berubah, aplikasi Anda akan dapat menerima dan menggunakan alamat IP baru sumber daya dengan meminta DNS.

Pada beberapa konfigurasi Java, TTL default JVM diatur sehingga tidak akan pernah menyegarkan entri DNS sampai JVM dimulai ulang. Jadi, jika alamat IP untuk AWS sumber daya berubah saat aplikasi Anda masih berjalan, itu tidak akan dapat menggunakan sumber daya itu sampai Anda

secara manual me-restart JVM dan informasi IP cache di-refresh. Dalam hal ini, sangat penting untuk mengatur TTL JVM sehingga secara berkala akan menyegarkan informasi IP cache.

### Note

TTL default dapat bervariasi sesuai dengan versi JVM Anda dan apakah [manajer keamanan diinstal](#). Banyak JVM menyediakan TTL bawaan yang kurang dari 60 detik. Jika Anda menggunakan JVM seperti itu dan tidak menggunakan manajer keamanan, Anda dapat mengabaikan sisa topik ini.

## Cara mengatur JVM TTL

Untuk memodifikasi TTL JVM, atur nilai properti [networkaddress.cache.ttl](#). Gunakan salah satu metode berikut, sesuai dengan kebutuhan Anda:

- Secara global, untuk semua aplikasi yang menggunakan JVM. Set `networkaddress.cache.ttl` dalam `$JAVA_HOME/jre/lib/security/java.security` file:

```
networkaddress.cache.ttl=60
```

- hanya untuk aplikasi Anda, atur `networkaddress.cache.ttl` dalam kode inisialisasi aplikasi Anda:

```
java.security.Security.setProperty("networkaddress.cache.ttl" , "60");
```

## Praktik terbaik AWS SDK for Java 2.x

Bagian ini mencantumkan praktik terbaik untuk menggunakan SDK for Java 2.x.

### Topik

- [Gunakan kembali klien SDK, jika memungkinkan](#)
- [Tutup aliran masukan dari operasi klien](#)
- [Menyetel konfigurasi HTTP berdasarkan tes kinerja](#)
- [Gunakan OpenSSL untuk klien HTTP berbasis Netty](#)

- [Konfigurasi batas waktu API](#)
- [Gunakan metrik](#)

## Gunakan kembali klien SDK, jika memungkinkan

Setiap klien SDK memelihara kumpulan koneksi HTTP sendiri. Koneksi yang sudah ada di kolam dapat digunakan kembali oleh permintaan baru untuk mengurangi waktu untuk membuat koneksi baru. Kami merekomendasikan berbagi satu contoh klien untuk menghindari overhead memiliki terlalu banyak kumpulan koneksi yang tidak digunakan secara efektif. Semua klien SDK aman untuk utas.

Jika Anda tidak ingin berbagi instance klien, panggil `close()` instance untuk melepaskan sumber daya saat klien tidak diperlukan.

## Tutup aliran masukan dari operasi klien

Untuk operasi streaming seperti [S3Client#getObject](#), jika Anda bekerja dengan [ResponseInputStream](#) langsung, kami sarankan Anda melakukan hal berikut:

- Baca semua data dari aliran input sesegera mungkin.
- Tutup aliran input sesegera mungkin.

Kami membuat rekomendasi ini karena aliran input adalah aliran langsung data dari koneksi HTTP dan koneksi HTTP yang mendasarinya tidak dapat digunakan kembali sampai semua data dari aliran telah dibaca dan aliran ditutup. Jika aturan ini tidak diikuti, klien dapat kehabisan sumber daya dengan mengalokasikan terlalu banyak koneksi HTTP yang terbuka, tetapi tidak terpakai.

## Menyetel konfigurasi HTTP berdasarkan tes kinerja

SDK menyediakan satu set [konfigurasi http default](#) yang berlaku untuk kasus penggunaan umum. Kami menyarankan agar pelanggan menyetel konfigurasi HTTP untuk aplikasi mereka berdasarkan kasus penggunaan mereka.

Sebagai titik awal yang baik, SDK menawarkan fitur [default konfigurasi cerdas](#). Fitur ini tersedia dimulai dengan versi 2.17.102. Anda memilih mode tergantung pada kasus penggunaan Anda, yang memberikan nilai konfigurasi yang masuk akal.

## Gunakan OpenSSL untuk klien HTTP berbasis Netty

Secara default, SDK [NettyNioAsyncHttpClient](#) menggunakan implementasi SSL default JDK sebagai `SslProvider`. Pengujian kami menemukan bahwa OpenSSL berkinerja lebih baik daripada implementasi default JDK. Komunitas Netty juga [merekomendasikan penggunaan OpenSSL](#).

Untuk menggunakan OpenSSL, `netty-tcnative` tambahkan dependensi Anda. Untuk detail konfigurasi, lihat [dokumentasi proyek Netty](#).

Setelah Anda `netty-tcnative` mengkonfigurasi untuk proyek Anda, `NettyNioAsyncHttpClient` instance akan secara otomatis memilih OpenSSL. Atau, Anda dapat mengatur secara `SslProvider` eksplisit menggunakan `NettyNioAsyncHttpClient` builder seperti yang ditunjukkan pada cuplikan berikut.

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

## Konfigurasi batas waktu API

SDK memberikan [nilai default](#) untuk beberapa opsi batas waktu, seperti batas waktu koneksi dan batas waktu socket, tetapi tidak untuk batas waktu panggilan API atau batas waktu percobaan panggilan API individual. Ini adalah praktik yang baik untuk mengatur batas waktu untuk upaya individu dan seluruh permintaan. Ini akan memastikan aplikasi Anda gagal dengan cepat dengan cara yang optimal ketika ada masalah sementara yang dapat menyebabkan upaya permintaan membutuhkan waktu lebih lama untuk menyelesaikan atau masalah jaringan yang fatal.

Anda dapat mengonfigurasi batas waktu untuk semua permintaan yang dibuat oleh klien layanan menggunakan [ClientOverrideConfiguration#apiCallAttemptTimeout](#) dan [ClientOverrideConfiguration#apiCallTimeout](#).

Contoh berikut menunjukkan konfigurasi klien Amazon S3 dengan nilai batas waktu kustom.

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

## apiCallAttemptTimeout

Pengaturan ini menetapkan jumlah waktu untuk satu percobaan HTTP, setelah itu panggilan API dapat dicoba ulang.

## apiCallTimeout

Nilai untuk properti ini mengonfigurasi jumlah waktu untuk seluruh eksekusi, termasuk semua upaya coba lagi.

Sebagai alternatif untuk mengatur nilai batas waktu ini pada klien layanan, Anda dapat menggunakan [RequestOverrideConfiguration#apiCallTimeout\(\)](#) dan [RequestOverrideConfiguration#apiCallAttemptTimeout\(\)](#) mengonfigurasi satu permintaan.

Contoh berikut mengonfigurasi `listBuckets` permintaan tunggal dengan nilai batas waktu kustom.

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

Saat Anda menggunakan properti ini bersama-sama, Anda menetapkan batas keras pada total waktu yang dihabiskan untuk semua upaya di seluruh percobaan ulang. Anda juga menetapkan permintaan HTTP individual untuk gagal cepat pada permintaan yang lambat.

## Gunakan metrik

SDK for Java [dapat mengumpulkan](#) metrik untuk klien layanan di aplikasi Anda. Anda dapat menggunakan metrik ini untuk mengidentifikasi masalah kinerja, meninjau tren penggunaan secara keseluruhan, meninjau pengecualian klien layanan yang dikembalikan, atau menggali untuk memahami masalah tertentu.

Kami menyarankan Anda mengumpulkan metrik, lalu menganalisis CloudWatch Log Amazon, untuk mendapatkan pemahaman yang lebih dalam tentang kinerja aplikasi Anda.

# Gunakan fitur AWS SDK for Java 2.x

## Fitur umum

SDK for Java 2.x berisi beberapa fitur yang membuat pemrograman Layanan AWS lebih mudah.

- [SDK menyembunyikan mekanisme kompleks di balik pengambilan hasil paginasi dan pemungutan suara untuk sumber daya.](#)
- [Pemrograman asinkron dengan I/O non-pemblokiran](#) membantu Anda menulis kode bersamaan dengan kinerja yang lebih baik. SDK memberikan manfaat [HTTP/2](#), seperti mengurangi latensi, jika memungkinkan.
- Java SDK dapat menghasilkan [metrik](#) untuk membantu Anda memantau kesehatan operasional aplikasi Anda.

## Fitur khusus layanan

Selain fitur umum yang disebutkan sebelumnya, Java SDK menyediakan fitur untuk spesifik Layanan AWS.

- Amazon S3 - Untuk [menyederhanakan pekerjaan Anda dengan file dan direktori dengan Amazon S3](#), SDK menyediakan S3 Transfer Manager. Untuk [meningkatkan kinerja dan keandalan](#) saat menggunakan API S3 asinkron standar SDK, SDK menawarkan klien S3 berbasis CRT. AWS
- DynamoDB - [Kemampuan pemetaan berorientasi objek](#) disediakan oleh DynamoDB Enhanced Client API. [Bekerja dengan data berorientasi dokumen bergaya JSON dengan menggunakan Enhanced Document API.](#)
- IAM - API Pembuat Kebijakan IAM menyediakan cara [berorientasi objek yang aman untuk](#) membuat kebijakan IAM.

## Bekerja dengan hasil paginasi menggunakan 2.x AWS SDK for Java

Banyak AWS operasi mengembalikan hasil paginasi ketika objek respons terlalu besar untuk dikembalikan dalam satu respons. Di AWS SDK for Java 1.0, respons berisi token yang Anda

gunakan untuk mengambil halaman hasil berikutnya. Sebaliknya, AWS SDK for Java 2.x memiliki metode autopagination yang membuat beberapa panggilan layanan untuk mendapatkan halaman hasil berikutnya untuk Anda secara otomatis. Anda hanya perlu menulis kode yang memproses hasilnya. Autopagination tersedia untuk klien sinkron dan asinkron.

### Note

Cuplikan kode ini mengasumsikan bahwa Anda memahami [dasar-dasar penggunaan SDK](#), dan telah mengonfigurasi lingkungan Anda dengan akses masuk [tunggal](#).

## pagination sinkron

Contoh berikut menunjukkan metode pagination sinkron untuk mencantumkan objek Amazon S3 dalam bucket.

### Iterasi di atas halaman

Contoh pertama menunjukkan penggunaan objek `ListRes` paginator, sebuah [ListObjectsV2Iterable](#) contoh, untuk iterasi melalui semua halaman respon dengan metode `stream`. Kode mengalir di atas halaman respons, mengubah aliran respons menjadi aliran [S3Object](#) konten, dan kemudian memproses konten objek. Amazon S3

Impor berikut berlaku untuk semua contoh di bagian pagination sinkron ini.

### Impor

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
```

```

import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;

```

```

ListObjectsV2Request listReq = ListObjectsV2Request.builder()
    .bucket(bucketName)
    .maxKeys(1)
    .build();

ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
// Process response pages
listRes.stream()
    .flatMap(r -> r.contents().stream())
    .forEach(content -> System.out
        .println(" Key: " + content.key() + "
size = " + content.size()));

```

Lihat [contoh lengkapnya](#) di GitHub.

## Iterasi di atas objek

Contoh berikut menunjukkan cara untuk mengulangi objek yang dikembalikan dalam respons, bukan halaman respons. `contents` Metode `ListObjectsV2Iterable` kelas mengembalikan sebuah [SdkIterable](#) yang menyediakan beberapa metode untuk memproses elemen konten yang mendasarinya.

Gunakan aliran

Cuplikan berikut menggunakan `stream` metode pada konten respons untuk mengulangi koleksi item paginasi.



```
// Helper method to work with paginated collection of items directly.
listRes.contents().stream()
    .forEach(content -> System.out
        .println(" Key: " + content.key() + "
size = " + content.size()));
```

Lihat [contoh lengkapnya](#) di GitHub.

Gunakan untuk setiap loop

Karena `SdkIterable` memperluas `Iterable` antarmuka, Anda dapat memproses konten seperti apa pun `Iterable`. Cuplikan berikut menggunakan `for-each` loop standar untuk iterasi melalui isi respon.

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

Lihat [contoh lengkapnya](#) di GitHub.

## Pagination manual

Jika kasus penggunaan Anda memerlukannya, pagination manual masih tersedia. Gunakan token berikutnya di objek respons untuk permintaan berikutnya. Contoh berikut menggunakan `while` loop.

```
ListObjectsV2Request listObjectsReqManual =
ListObjectsV2Request.builder()
    .bucket(bucketName)
    .maxKeys(1)
    .build();

boolean done = false;
while (!done) {
    ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
    for (S3Object content : listObjResponse.contents()) {
        System.out.println(content.key());
    }
}
```

```
        if (listObjResponse.nextContinuationToken() == null) {
            done = true;
        }

        listObjectsReqManual = listObjectsReqManual.toBuilder()

        .continuationToken(listObjResponse.nextContinuationToken())
            .build();
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## pagination asinkron

Contoh berikut menunjukkan metode pagination asinkron untuk daftar tabel. DynamoDB

### Iterasi di atas halaman nama tabel

Dua contoh berikut menggunakan klien DynamoDB asinkron yang memanggil metode dengan permintaan `listTablesPaginator` untuk mendapatkan file. [ListTablesPublisher](#) `ListTablesPublisher` mengimplementasikan dua antarmuka, yang menyediakan banyak opsi untuk memproses respons. Kita akan melihat metode masing-masing antarmuka.

### Gunakan **Subscriber**

Contoh kode berikut menunjukkan bagaimana memproses hasil paginasi dengan menggunakan `org.reactivestreams.Publisher` antarmuka yang diimplementasikan oleh

`ListTablesPublisher` Untuk mempelajari lebih lanjut tentang model aliran reaktif, lihat repo [Reactive Streams](#). GitHub

Impor berikut berlaku untuk semua contoh di bagian pagination asinkron ini.

### Impor

```
import io.reactivex.rxjava3.core.Flowable;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import reactor.core.publisher.Flux;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
```

```
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.paginators.ListTablesPublisher;

import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
```

Kode berikut memperoleh sebuah `ListTablesPublisher` instance.

```
// Creates a default client with credentials and region loaded from the
// environment.
final DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();

ListTablesRequest listTablesRequest =
ListTablesRequest.builder().limit(3).build();
ListTablesPublisher publisher =
asyncClient.listTablesPaginator(listTablesRequest);
```

Kode berikut menggunakan implementasi anonim `org.reactivestreams.Subscriber` untuk memproses hasil untuk setiap halaman.

`onSubscribe` Metode ini memanggil `Subscription.request` metode untuk memulai permintaan data dari penerbit. Metode ini harus dipanggil untuk mulai mendapatkan data dari penerbit.

`onNext` Metode pelanggan memproses halaman respons dengan mengakses semua nama tabel dan mencetak masing-masing. Setelah halaman diproses, halaman lain diminta dari penerbit. Metode ini dipanggil berulang kali sampai semua halaman diambil.

`onError` Metode ini dipicu jika terjadi kesalahan saat mengambil data. Akhirnya, `onComplete` metode ini dipanggil ketika semua halaman telah diminta.

```
// A Subscription represents a one-to-one life-cycle of a Subscriber
subscribing
// to a Publisher.
publisher.subscribe(new Subscriber<ListTablesResponse>() {
    // Maintain a reference to the subscription object, which is required to
request
    // data from the publisher.
    private Subscription subscription;

    @Override
    public void onSubscribe(Subscription s) {
```

```

        subscription = s;
        // Request method should be called to demand data. Here we request a
single
        // page.
        subscription.request(1);
    }

    @Override
    public void onNext(ListTablesResponse response) {
        response.tableNames().forEach(System.out::println);
        // After you process the current page, call the request method to
signal that
        // you are ready for next page.
        subscription.request(1);
    }

    @Override
    public void onError(Throwable t) {
        // Called when an error has occurred while processing the requests.
    }

    @Override
    public void onComplete() {
        // This indicates all the results are delivered and there are no more
pages
        // left.
    }
});

```

Lihat [contoh lengkapnya](#) di GitHub.

## Gunakan **Consumer**

`SdkPublisherAntarmuka` yang `ListTablesPublisher` mengimplementasikan memiliki `subscribe` metode yang mengambil `Consumer` dan mengembalikan `aCompletableFuture<Void>`.

`subscribeMetode` dari antarmuka ini dapat digunakan untuk kasus penggunaan sederhana ketika overhead `org.reactivestreams.Subscriber` mungkin terlalu banyak. Karena kode di bawah ini menghabiskan setiap halaman, ia memanggil `tableNames` metode pada masing-masing halaman. `tableNamesMetode` mengembalikan nama tabel DynamoDB yang diproses dengan metode `java.util.List.forEach`

```
// Use a Consumer for simple use cases.
CompletableFuture<Void> future = publisher.subscribe(
    response -> response.tableNames()
        .forEach(System.out::println));
```

Lihat [contoh lengkapnya](#) di GitHub.

## Ulangi nama tabel

Contoh berikut menunjukkan cara untuk mengulangi objek yang dikembalikan dalam respons, bukan halaman respons. Mirip dengan contoh Amazon S3 sinkron yang sebelumnya ditampilkan dengan contents metodenya, kelas hasil asinkron `DynamoDBListTablesPublisher`, memiliki `tableNames` metode kenyamanan untuk berinteraksi dengan koleksi item yang mendasarinya. Jenis pengembalian `tableNames` metode adalah [SdkPublisher](#) yang dapat digunakan untuk meminta item di semua halaman.

## Gunakan **Subscriber**

Kode berikut memperoleh koleksi `SdkPublisher` yang mendasari nama tabel.

```
// Create a default client with credentials and region loaded from the
// environment.
final DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();

ListTablesRequest listTablesRequest =
ListTablesRequest.builder().limit(3).build();
ListTablesPublisher listTablesPublisher =
asyncClient.listTablesPaginator(listTablesRequest);
SdkPublisher<String> publisher = listTablesPublisher.tableNames();
```

Kode berikut menggunakan implementasi anonim `org.reactivestreams.Subscriber` untuk memproses hasil untuk setiap halaman.

`onNext` Metode pelanggan memproses elemen individual dari koleksi. Dalam hal ini, itu adalah nama tabel. Setelah nama tabel diproses, nama tabel lain diminta dari penerbit. Metode ini dipanggil berulang kali sampai semua nama tabel diambil.

```
// Use a Subscriber.
publisher.subscribe(new Subscriber<String>() {
    private Subscription subscription;
```

```
@Override
public void onSubscribe(Subscription s) {
    subscription = s;
    subscription.request(1);
}

@Override
public void onNext(String tableName) {
    System.out.println(tableName);
    subscription.request(1);
}

@Override
public void onError(Throwable t) {
}

@Override
public void onComplete() {
}
});
```

Lihat [contoh lengkapnya](#) di GitHub.

## Gunakan **Consumer**

Contoh berikut menggunakan subscribe metode SdkPublisher yang membutuhkan a Consumer untuk memproses setiap item.

```
// Use a Consumer.
CompletableFuture<Void> future = publisher.subscribe(System.out::println);
future.get();
```

Lihat [contoh lengkapnya](#) di GitHub.

## Gunakan pustaka pihak ketiga

Anda dapat menggunakan pustaka pihak ketiga lainnya alih-alih mengimplementasikan pelanggan khusus. Contoh ini menunjukkan penggunaan RxJava, tetapi pustaka apa pun yang mengimplementasikan antarmuka aliran reaktif dapat digunakan. Lihat [halaman RxJava wiki GitHub](#) untuk informasi lebih lanjut tentang perpustakaan itu.

Untuk menggunakan perpustakaan, tambahkan sebagai dependensi. Jika menggunakan Maven, contoh menunjukkan cuplikan POM untuk digunakan.

## Entri POM

```
<dependency>
  <groupId>io.reactivex.rxjava3</groupId>
  <artifactId>rxjava</artifactId>
  <version>3.1.6</version>
</dependency>
```

## Kode

```
DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();
ListTablesPublisher publisher =
asyncClient.listTablesPaginator(ListTablesRequest.builder()
    .build());

// The Flowable class has many helper methods that work with
// an implementation of an org.reactivestreams.Publisher.
List<String> tables = Flowable.fromPublisher(publisher)
    .flatMapIterable(ListTablesResponse::tableNames)
    .toList()
    .blockingGet();
System.out.println(tables);
```

Lihat [contoh lengkapnya](#) di GitHub.

## Polling untuk status sumber daya di AWS SDK for Java 2.x:

### Pelayan

Utilitas pelayan dari AWS SDK for Java 2.x memungkinkan Anda untuk memvalidasi bahwa AWS sumber daya berada dalam keadaan tertentu sebelum melakukan operasi pada sumber daya tersebut.

Pelayan adalah abstraksi yang digunakan untuk polling AWS sumber daya, seperti DynamoDB tabel atau Amazon S3 ember, sampai keadaan yang diinginkan tercapai (atau sampai penentuan dibuat bahwa sumber daya tidak akan pernah mencapai keadaan yang diinginkan). Alih-alih menulis logika untuk terus polling AWS sumber daya Anda, yang dapat menjadi rumit dan rawan kesalahan, Anda

dapat menggunakan pelayan untuk polling sumber daya dan membuat kode Anda terus berjalan setelah sumber daya siap.

## Prasyarat

Sebelum Anda dapat menggunakan pelayan dalam proyek dengan AWS SDK for Java, Anda harus menyelesaikan langkah-langkah dalam [Menyiapkan 2.x](#). AWS SDK for Java

Anda juga harus mengonfigurasi dependensi proyek Anda (misalnya, di `build.gradle` file `pom.xml` atau Anda) untuk menggunakan versi `2.15.0` atau versi yang lebih baru. AWS SDK for Java

Sebagai contoh:

```
<project>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.15.0</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>
```

## Menggunakan pelayan

Untuk membuat instance objek pelayan, pertama buat klien layanan. Tetapkan `waiter()` metode klien layanan sebagai nilai objek pelayan. Setelah instance pelayan ada, atur opsi responsnya untuk mengeksekusi kode yang sesuai.

### Pemrograman sinkron

Cuplikan kode berikut menunjukkan cara menunggu DynamoDB tabel ada dan berada dalam status AKTIF.

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



```
DynamoDbWaiter waiter = dynamo.waiter();

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

// print out the matched response with a tableStatus of ACTIVE
waiterResponse.matched().response().ifPresent(System.out::println);
```

## Pemrograman asinkron

Cuplikan kode berikut menunjukkan cara menunggu DynamoDB tabel tidak ada lagi.

```
DynamoDbAsyncClient asyncDynamo = DynamoDbAsyncClient.create();
DynamoDbAsyncWaiter asyncWaiter = asyncDynamo.waiter();

CompletableFuture<WaiterResponse<DescribeTableResponse>> waiterResponse =
    asyncWaiter.waitUntilTableNotExists(r -> r.tableName("myTable"));

waiterResponse.whenComplete((r, t) -> {
    if (t == null) {
        // print out the matched ResourceNotFoundException
        r.matched().exception().ifPresent(System.out::println);
    }
}).join();
```

## Konfigurasi pelayan

Anda dapat menyesuaikan konfigurasi untuk pelayan dengan menggunakan `overrideConfiguration()` pada pembangunnya. Untuk beberapa operasi, Anda dapat menerapkan konfigurasi khusus saat Anda membuat permintaan.

### Konfigurasi pelayan

Cuplikan kode berikut menunjukkan cara mengganti konfigurasi pada pelayan.

```
// sync
DynamoDbWaiter waiter =
    DynamoDbWaiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(10))
        .client(dynamoDbClient)
        .build();
```

```
// async
DynamoDbAsyncWaiter asyncWaiter =
    DynamoDbAsyncWaiter.builder()
        .client(dynamoDbAsyncClient)
        .overrideConfiguration(o -> o.backoffStrategy(
            FixedDelayBackoffStrategy.create(Duration.ofSeconds(2))))
        .scheduledExecutorService(Executors.newScheduledThreadPool(3))
        .build();
```

## Ganti konfigurasi untuk permintaan tertentu

Cuplikan kode berikut menunjukkan cara mengganti konfigurasi untuk pelayan berdasarkan permintaan. Perhatikan bahwa hanya beberapa operasi yang memiliki konfigurasi yang dapat disesuaikan.

```
waiter.waitForTableExists(b -> b.tableName("myTable"),
    o -> o.maxAttempts(10));

asyncWaiter.waitForTableExists(b -> b.tableName("myTable"),
    o -> o.waitFor(Duration.ofMinutes(1)));
```

## Contoh kode

Untuk contoh lengkap menggunakan pelayan dengan DynamoDB, lihat [CreateTable.java](#) di Repositori Contoh AWS Kode.

Untuk contoh lengkap menggunakan pelayan dengan Amazon S3, lihat [S3 BucketOps.java](#) di Repositori Contoh Kode. AWS

## Gunakan pemrograman asinkron

AWS SDK for Java 2.x Fitur klien asinkron dengan dukungan I/O non-pemblokiran yang menerapkan konkurensi tinggi di beberapa utas. Namun, total I/O non-pemblokiran tidak dijamin. Klien asinkron dapat melakukan pemblokiran panggilan dalam beberapa kasus seperti pengambilan kredensial, penandatanganan permintaan menggunakan [AWS Sigv4 \(SigV4\)](#), atau penemuan titik akhir.

Metode sinkron memblokir eksekusi thread Anda hingga klien menerima respons dari layanan. Metode asinkron segera kembali, memberikan kontrol kembali ke utas panggilan tanpa menunggu respons.

Karena metode asinkron kembali sebelum respons tersedia, Anda memerlukan cara untuk mendapatkan respons saat sudah siap. Metode untuk klien asinkron di 2.x dari `CompletableFuture` objek AWS SDK for Java kembali yang memungkinkan Anda mengakses respons saat sudah siap.

## Operasi non-streaming

Untuk operasi non-streaming, panggilan metode asinkron mirip dengan metode sinkron. Namun, metode asinkron dalam AWS SDK for Java mengembalikan [CompletableFuture](#) objek yang berisi hasil operasi asinkron di masa depan.

Panggil `CompletableFuture whenComplete()` metode dengan tindakan untuk menyelesaikan ketika hasilnya tersedia. `CompletableFuture` mengimplementasikan `Future` antarmuka, sehingga Anda juga bisa mendapatkan objek respons dengan memanggil `get()` metode.

Berikut ini adalah contoh operasi asinkron yang memanggil Amazon DynamoDB fungsi untuk mendapatkan daftar tabel, menerima `CompletableFuture` yang dapat menampung objek. [ListTablesResponse](#) Tindakan yang ditentukan dalam panggilan ke `whenComplete()` dilakukan hanya ketika panggilan asinkron selesai.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;
import java.util.concurrent.CompletableFuture;
```

### Kode

```
public class DynamoDBAsyncListTables {

    public static void main(String[] args) throws InterruptedException {

        // Create the DynamoDbAsyncClient object
        Region region = Region.US_EAST_1;
        DynamoDbAsyncClient client = DynamoDbAsyncClient.builder()
            .region(region)
            .build();

        listTables(client);
    }
}
```

```
}

public static void listTables(DynamoDbAsyncClient client) {

    CompletableFuture<ListTablesResponse> response =
client.listTables(ListTablesRequest.builder()
        .build());

    // Map the response to another CompletableFuture containing just the table
names
    CompletableFuture<List<String>> tableNames =
response.thenApply(ListTablesResponse::tableNames);

    // When future is complete (either successfully or in error) handle the
response
    tableNames.whenComplete((tables, err) -> {
        try {
            if (tables != null) {
                tables.forEach(System.out::println);
            } else {
                // Handle error
                err.printStackTrace();
            }
        } finally {
            // Lets the application shut down. Only close the client when you are
completely done with it.
            client.close();
        }
    });
    tableNames.join();
}
}
```

Contoh kode berikut menunjukkan cara mengambil Item dari tabel dengan menggunakan klien Asynchronous. Memanggil `getItem` metode `DynamoDbAsyncClient` dan meneruskannya [GetItemRequest](#) objek dengan nama tabel dan nilai kunci utama dari item yang Anda inginkan. Ini biasanya bagaimana Anda meneruskan data yang diperlukan operasi. Dalam contoh ini, perhatikan bahwa nilai String dilewatkan.

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
```

```
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

## Kode

```
public static void getItem(DynamoDbAsyncClient client, String tableName, String
key, String keyVal) {

    HashMap<String, AttributeValue> keyToGet =
        new HashMap<String, AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal).build());

    try {

        // Create a GetItemRequest instance
        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName(tableName)
            .build();

        // Invoke the DynamoDbAsyncClient object's getItem
        java.util.Collection<AttributeValue> returnedItem =
client.getItem(request).join().item().values();

        // Convert Set to Map
        Map<String, AttributeValue> map =
returnedItem.stream().collect(Collectors.toMap(AttributeValue::s, s->s));
        Set<String> keys = map.keySet();
        for (String sinKey : keys) {
            System.out.format("%s: %s\n", sinKey, map.get(sinKey).toString());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Operasi streaming

Untuk operasi streaming, Anda harus menyediakan [AsyncRequestBody](#) untuk menyediakan konten secara bertahap, atau [AsyncResponseTransformer](#) untuk menerima dan memproses respons.

Contoh berikut mengunggah file ke Amazon S3 asinkron dengan menggunakan operasi `PutObject`

### Impor

```
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import java.nio.file.Paths;
import java.util.concurrent.CompletableFuture;
```

### Kode

```
/**
 * To run this AWS code example, ensure that you have setup your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class S3AsyncOps {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "Usage:\n" +
            "    S3AsyncOps <bucketName> <key> <path>\n\n" +
            "Where:\n" +
            "    bucketName - the name of the Amazon S3 bucket (for example,
bucket1). \n\n" +
            "    key - the name of the object (for example, book.pdf). \n" +
            "    path - the local path to the file (for example, C:/AWS/book.pdf).
\n" ;
```

```
    if (args.length != 3) {
        System.out.println(USAGE);
        System.exit(1);
    }

    String bucketName = args[0];
    String key = args[1];
    String path = args[2];

    Region region = Region.US_WEST_2;
    S3AsyncClient client = S3AsyncClient.builder()
        .region(region)
        .build();

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

    // Put the object into the bucket
    CompletableFuture<PutObjectResponse> future = client.putObject(objectRequest,
        AsyncRequestBody.fromFile(Paths.get(path))
    );
    future.whenComplete((resp, err) -> {
        try {
            if (resp != null) {
                System.out.println("Object uploaded. Details: " + resp);
            } else {
                // Handle error
                err.printStackTrace();
            }
        } finally {
            // Only close the client when you are completely done with it
            client.close();
        }
    });

    future.join();
}
```

Contoh berikut mendapatkan file dari Amazon S3 asinkron dengan menggunakan operasi `GetObject`

## Impor

```
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.nio.file.Paths;
import java.util.concurrent.CompletableFuture;
```

## Kode

```
/**
 * To run this AWS code example, ensure that you have setup your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class S3AsyncStreamOps {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "Usage:\n" +
            "  S3AsyncStreamOps <bucketName> <objectKey> <path>\n\n" +
            "Where:\n" +
            "  bucketName - the name of the Amazon S3 bucket (for example,
bucket1). \n\n" +
            "  objectKey - the name of the object (for example, book.pdf). \n" +
            "  path - the local path to the file (for example, C:/AWS/book.pdf).
\n" ;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }
    }
}
```



```
String bucketName = args[0];
String objectKey = args[1];
String path = args[2];

Region region = Region.US_WEST_2;
S3AsyncClient client = S3AsyncClient.builder()
    .region(region)
    .build();

GetObjectRequest objectRequest = GetObjectRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .build();

CompletableFuture<GetObjectResponse> futureGet =
client.getObject(objectRequest,
    AsyncResponseTransformerToFile(Paths.get(path)));

futureGet.whenComplete((resp, err) -> {
    try {
        if (resp != null) {
            System.out.println("Object downloaded. Details: "+resp);
        } else {
            err.printStackTrace();
        }
    } finally {
        // Only close the client when you are completely done with it
        client.close();
    }
});
futureGet.join();
}
```

## Operasi lanjutan

AWS SDK for Java 2.x menggunakan [Netty](#), kerangka kerja aplikasi jaringan berbasis peristiwa asinkron, untuk menangani utas I/O. AWS SDK for Java 2.x membuat Netty ExecutorService di belakang, untuk menyelesaikan futures yang dikembalikan dari permintaan klien HTTP hingga ke klien Netty. Abstraksi ini mengurangi risiko aplikasi merusak proses asinkron jika pengembang memilih untuk menghentikan atau menidurkan utas. Secara default, setiap klien asinkron membuat

threadpool berdasarkan jumlah prosesor dan mengelola tugas dalam antrian di dalam file.

## ExecutorService

Pengguna tingkat lanjut dapat menentukan ukuran kumpulan utas mereka saat membuat klien asinkron menggunakan opsi berikut saat membuat.

### Kode

```
S3AsyncClient clientThread = S3AsyncClient.builder()
    .asyncConfiguration(
        b -> b.advancedOption(SdkAdvancedAsyncClientOption
            .FUTURE_COMPLETION_EXECUTOR,
            Executors.newFixedThreadPool(10)
        )
    )
    .build();
```

Untuk mengoptimalkan kinerja, Anda dapat mengelola pelaksana kumpulan utas Anda sendiri, dan menyertakannya saat mengonfigurasi klien Anda.

```
ThreadPoolExecutor executor = new ThreadPoolExecutor(50, 50,
    10, TimeUnit.SECONDS,
    new LinkedBlockingQueue<>(<custom_value>),
    new ThreadFactoryBuilder()
        .threadNamePrefix("sdk-async-response").build());

// Allow idle core threads to time out
executor.allowCoreThreadTimeOut(true);

S3AsyncClient clientThread = S3AsyncClient.builder()
    .asyncConfiguration(
        b -> b.advancedOption(SdkAdvancedAsyncClientOption
            .FUTURE_COMPLETION_EXECUTOR,
            executor
        )
    )
    .build();
```

## Bekerja dengan HTTP/2 di AWS SDK for Java

HTTP/2 adalah revisi besar dari protokol HTTP. Versi baru ini memiliki beberapa peningkatan untuk meningkatkan kinerja:

- Pengkodean data biner memberikan transfer data yang lebih efisien.
- Kompresi header mengurangi byte overhead yang diunduh oleh klien, membantu mendapatkan konten ke klien lebih cepat. Ini sangat berguna untuk klien seluler yang sudah dibatasi bandwidth.
- Komunikasi asinkron dua arah (multiplexing) memungkinkan beberapa permintaan dan pesan respons antara klien dan AWS berada dalam penerbangan pada saat yang sama melalui satu koneksi, bukan melalui beberapa koneksi, yang meningkatkan kinerja.

Pengembang yang memutakhirkan ke SDK terbaru akan secara otomatis menggunakan HTTP/2 ketika didukung oleh layanan yang mereka kerjakan. Antarmuka pemrograman baru dengan mulus memanfaatkan fitur HTTP/2 dan menyediakan cara baru untuk membangun aplikasi.

AWS SDK for Java 2.x menampilkan API baru untuk streaming acara yang mengimplementasikan protokol HTTP/2. Untuk contoh cara menggunakan API baru ini, lihat [Bekerja dengan Kinesis](#).

## Gunakan metrik SDK dari AWS SDK for Java

Dengan AWS SDK for Java 2.x, Anda dapat mengumpulkan metrik tentang klien layanan di aplikasi Anda, menganalisis output Amazon CloudWatch, dan kemudian menindaklanjutinya.

Secara default, koleksi metrik dinonaktifkan di SDK. Topik ini membantu Anda mengaktifkan dan mengonfigurasinya.

## Prasyarat

Sebelum Anda dapat mengaktifkan dan menggunakan metrik, Anda harus menyelesaikan langkah-langkah berikut:

- Selesaikan langkah-langkah dalam [Pengaturan](#).
- Konfigurasikan dependensi proyek Anda (misalnya, di `build.gradle` file `pom.xml` atau Anda) untuk menggunakan versi `2.14.0` atau versi yang lebih baru. AWS SDK for Java

Untuk mengaktifkan penerbitan metrik CloudWatch, sertakan juga ArtifactID `ccloudwatch-metric-publisher` dengan nomor `2.14.0` versi atau yang lebih baru di dependensi proyek Anda.

Sebagai contoh:

```
<project>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.14.0</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>cloudwatch-metric-publisher</artifactId>
      <version>2.14.0</version>
    </dependency>
  </dependencies>
</project>
```

- Aktifkan `ccloudwatch:PutMetricData` izin untuk identitas IAM untuk memungkinkan SDK for Java menulis metrik.

## Cara mengaktifkan pengumpulan metrik

Anda dapat mengaktifkan metrik dalam aplikasi Anda untuk klien layanan atau permintaan individual.

### Aktifkan metrik untuk permintaan tertentu

Cuplikan kode berikut menunjukkan cara mengaktifkan penerbit CloudWatch metrik untuk permintaan. Amazon DynamoDB Ini menggunakan konfigurasi penerbit metrik default.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();
```

```
DynamoDbClient ddb = DynamoDbClient.create();
ddb.listTables(ListTablesRequest.builder()
    .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
    .build());
```

## Aktifkan metrik untuk klien layanan tertentu

Cuplikan kode berikut menunjukkan cara mengaktifkan penerbit CloudWatch metrik dengan pengaturan default untuk klien layanan.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();

DynamoDbClient ddb = DynamoDbClient.builder()
    .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
    .build();
```

Cuplikan berikut menunjukkan cara menggunakan konfigurasi kustom untuk penerbit metrik untuk klien layanan tertentu. Kustomisasi termasuk memuat profil kredensial tertentu, menentukan wilayah yang berbeda dari klien layanan, dan menyesuaikan seberapa sering penerbit mengirimkan metrik. CloudWatch

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.builder()
    .cloudWatchClient(CloudWatchAsyncClient.builder()
        .region(Region.US_WEST_2)

        .credentialsProvider(ProfileCredentialsProvider.create("cloudwatch"))
        .build())

    .uploadFrequency(Duration.ofMinutes(5))
    .build();

DynamoDbClient ddb = DynamoDbClient.builder()
    .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
    .build();
```

## Kapan metrik tersedia?

Metrik umumnya tersedia dalam waktu 5-10 menit setelah SDK for Java memancarkannya. Untuk akurasi dan up-to-date metrik, periksa Cloudwatch setidaknya 10 menit setelah memancarkan metrik dari aplikasi Java Anda.

## Informasi apa yang dikumpulkan?

Koleksi metrik meliputi:

- Jumlah permintaan API, termasuk apakah mereka berhasil atau gagal
- Informasi tentang AWS layanan yang Anda panggil dalam permintaan API, termasuk pengecualian yang dikembalikan
- Durasi untuk berbagai operasi seperti Marshalling, Signing, dan permintaan HTTP
- Metrik klien HTTP, seperti jumlah koneksi terbuka, jumlah permintaan yang tertunda, dan nama klien HTTP yang digunakan

### Note

Metrik yang tersedia bervariasi menurut klien HTTP.

Untuk daftar lengkapnya, lihat [Metrik klien layanan](#).

## Bagaimana saya bisa menggunakan informasi ini?

Anda dapat menggunakan metrik yang dikumpulkan SDK untuk memantau klien layanan dalam aplikasi Anda. Anda dapat melihat tren penggunaan secara keseluruhan, mengidentifikasi anomali, meninjau pengecualian klien layanan yang dikembalikan, atau menggali untuk memahami masalah tertentu. Dengan menggunakan Amazon CloudWatch, Anda juga dapat membuat alarm untuk memberi tahu Anda segera setelah aplikasi Anda mencapai kondisi yang Anda tentukan.

Untuk informasi selengkapnya, lihat [Menggunakan Amazon CloudWatch Metrik](#) dan [Menggunakan Amazon CloudWatch Alarm](#) di [Amazon CloudWatch Panduan Pengguna](#).

## Metrik klien layanan

[Dengan itu AWS SDK for Java 2.x, Anda dapat mengumpulkan metrik dari klien layanan di aplikasi Anda dan kemudian menerbitkan \(mengeluarkan\) metrik tersebut ke Amazon. CloudWatch](#)

Tabel ini mencantumkan metrik yang dapat Anda kumpulkan dan persyaratan penggunaan klien HTTP apa pun.

[Untuk informasi selengkapnya tentang mengaktifkan dan mengonfigurasi metrik untuk SDK, lihat Mengaktifkan metrik SDK.](#)

Istilah yang digunakan dalam tabel berarti:

- Apache: klien HTTP berbasis Apache () [ApacheHttpClient](#)
- Netty: klien HTTP berbasis Netty () [NettyNioAsyncHttpClient](#)
- CRT: klien HTTP AWS berbasis CRT () [AwsCrtAsyncHttpClient](#)
- Setiap: pengumpulan data metrik tidak bergantung pada klien HTTP; ini termasuk penggunaan klien HTTP berbasis URLConnection () [URLConnectionHttpClient](#)

Metrik dikumpulkan dengan setiap permintaan

Nama metrik	Deskripsi	Jenis	Klien HTTP diperlukan
ApiCallDuration	Total waktu yang dibutuhkan untuk menyelesaikan permintaan (termasuk semua percobaan ulang)	Durasi	Sebarang
ApiCallSuccessful	Benar jika panggilan API berhasil; false jika tidak	Boolean	Sebarang
CredentialsFetchDuration	Waktu yang dibutuhkan untuk mengambil kredensial AWS penandatanganan untuk permintaan	Durasi	Sebarang
MarshallingDuration	Waktu yang dibutuhkan untuk memenuhi permintaan	Durasi	Sebarang
OperationName	Nama AWS API permintaan dibuat untuk	String	Sebarang

Nama metrik	Deskripsi	Jenis	Klien HTTP diperlukan
RetryCount	Berapa kali SDK mencoba ulang panggilan API	Bilangan Bulat	Sebarang
ServiceId	ID layanan dari permintaan API Layanan AWS yang dibuat terhadap	String	Sebarang
TokenFetchDuration	Waktu yang dibutuhkan untuk mengambil kredensial penandatangan token untuk permintaan tersebut	Durasi	Sebarang

## Metrik dikumpulkan untuk setiap upaya permintaan

Setiap panggilan API mungkin memerlukan beberapa upaya sebelum respons diterima. Metrik ini dikumpulkan untuk setiap upaya.

Nama metrik	Deskripsi	Jenis	Klien HTTP diperlukan
AvailableConcurrency	Jumlah permintaan bersamaan yang tersisa yang dapat didukung oleh klien HTTP tanpa perlu membuat koneksi lain	Bilangan Bulat	Apache, Netty, CRT
AwsExtendedRequestId	ID permintaan yang diperpanjang dari permintaan layanan	String	Sebarang



Nama metrik	Deskripsi	Jenis	Klien HTTP diperlukan
AwsRequestId	ID permintaan permintaan layanan	String	Sebarang
BackoffDelayDuration	Durasi waktu SDK menunggu sebelum upaya panggilan API ini	Durasi	Sebarang
ConcurrencyAcquireDuration	Waktu yang dibutuhkan untuk mendapatkan saluran dari kolam koneksi	Durasi	Apache, Netty, CRT
HttpClientName	Nama HTTP yang digunakan untuk permintaan	String	Apache, Netty, CRT
HttpStatusCode	Kode status dikembalikan dengan respon HTTP	Bilangan Bulat	Sebarang
LeasedConcurrency	Jumlah permintaan yang saat ini sedang dijalankan oleh klien HTTP	Bilangan Bulat	Apache, Netty, CRT
LocalStreamWindowSize	Ukuran jendela HTTP/2 lokal dalam byte untuk aliran tempat permintaan ini dijalankan	Bilangan Bulat	Netty

Nama metrik	Deskripsi	Jenis	Klien HTTP diperlukan
MarshallingDuration	Waktu yang dibutuhkan untuk mengirim permintaan SDK ke permintaan HTTP	Durasi	Sebarang
MaxConcurrency	Jumlah maksimum permintaan bersamaan yang didukung oleh klien HTTP	Bilangan Bulat	Apache, Netty, CRT
PendingConcurrencyAcquires	Jumlah permintaan yang diblokir, menunggu koneksi TCP lain atau aliran baru tersedia dari kumpulan koneksi	Bilangan Bulat	Apache, Netty, CRT
RemoteStreamWindowSize	Ukuran jendela HTTP/2 jarak jauh dalam byte untuk aliran tempat permintaan ini dijalankan	Bilangan Bulat	Netty
ServiceCallDuration	Waktu yang diperlukan untuk terhubung ke layanan, mengirim permintaan, dan menerima kode status HTTP dan header dari respons	Durasi	Sebarang

Nama metrik	Deskripsi	Jenis	Klien HTTP diperlukan
SigningDuration	Waktu yang dibutuhkan untuk menandatangani permintaan HTTP	Durasi	Sebarang
UnmarshallingDuration	Waktu yang dibutuhkan untuk membatalkan respons HTTP terhadap respons SDK	Durasi	Sebarang

# Bekerja dengan Layanan AWS menggunakan AWS SDK for Java 2.x

Bagian ini memberikan tutorial singkat dan panduan tentang cara bekerja dengan pilih Layanan AWS. Untuk kumpulan contoh lengkap, lihat [bagian Contoh Kode](#).

## Topik

- [Bekerja dengan CloudWatch](#)
- [AWSlayanan database dan AWS SDK for Java 2.x](#)
- [Bekerja dengan DynamoDB](#)
- [Bekerja dengan Amazon EC2](#)
- [Bekerja dengan IAM](#)
- [Bekerja dengan Kinesis](#)
- [Memanggil, membuat daftar, dan menghapus fungsi AWS Lambda](#)
- [Bekerja dengan Amazon S3](#)
- [Bekerja dengan Amazon Simple Notification Service](#)
- [Bekerja dengan Amazon Simple Queue Service](#)
- [Bekerja dengan Amazon Transcribe](#)

## Bekerja dengan CloudWatch

Bagian ini memberikan contoh pemrograman [Amazon CloudWatch](#) dengan menggunakan AWS SDK for Java 2.x.

Amazon CloudWatchmemantau sumber daya Amazon Web Services (AWS) Anda dan aplikasi yang Anda jalankan AWS secara real time. Anda dapat menggunakan CloudWatch untuk mengumpulkan dan melacak metrik, yang merupakan variabel yang dapat Anda ukur untuk sumber daya dan aplikasi Anda. CloudWatchalarm mengirim pemberitahuan atau secara otomatis membuat perubahan pada sumber daya yang Anda pantau berdasarkan aturan yang Anda tetapkan.

Contoh berikut hanya mencakup kode yang diperlukan untuk mendemonstrasikan setiap teknik. [Kode contoh lengkap tersedia di GitHub](#). Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk dibangun dan dijalankan.

## Topik

- [Dapatkan metrik dari CloudWatch](#)
- [Publikasikan data metrik kustom ke CloudWatch](#)
- [Bekerja dengan CloudWatch alarm](#)
- [Gunakan CloudWatch Acara Amazon](#)

## Dapatkan metrik dari CloudWatch

### Metrik daftar

Untuk membuat daftar CloudWatch metrik, buat [ListMetricsRequest](#) dan panggil `listMetrics` metode `CloudWatchClient` ini. Anda dapat menggunakan `ListMetricsRequest` untuk memfilter metrik yang dikembalikan berdasarkan namespace, nama metrik, atau dimensi.

#### Note

Daftar metrik dan dimensi yang diposting oleh AWS layanan dapat ditemukan dalam [Referensi Amazon CloudWatch Metrik dan Dimensi](#) di Amazon CloudWatch Panduan Pengguna.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
```

### Kode

```
public static void listMets( CloudWatchClient cw, String namespace) {

    boolean done = false;
    String nextToken = null;

    try {
```

```
while(!done) {

    ListMetricsResponse response;

    if (nextToken == null) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        response = cw.listMetrics(request);
    } else {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .nextToken(nextToken)
            .build();

        response = cw.listMetrics(request);
    }

    for (Metric metric : response.metrics()) {
        System.out.printf(
            "Retrieved metric %s", metric.metricName());
        System.out.println();
    }

    if(response.nextToken() == null) {
        done = true;
    } else {
        nextToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Metrik dikembalikan dalam a [ListMetricsResponse](#) dengan memanggil `getMetrics` metodenya.

Hasilnya mungkin paged. Untuk mengambil kumpulan hasil berikutnya, panggil `nextToken` objek respons dan gunakan nilai token untuk membangun objek permintaan baru. Kemudian panggil `listMetrics` metode lagi dengan permintaan baru.

Lihat [contoh lengkapnya](#) di GitHub.

## Informasi lain

- [ListMetrics](#) di Referensi Amazon CloudWatch API

## Publikasikan data metrik kustom ke CloudWatch

Sejumlah AWS layanan mempublikasikan [metrik mereka sendiri](#) di ruang nama yang dimulai dengan "AWS" Anda juga dapat mempublikasikan data metrik khusus menggunakan namespace Anda sendiri (asalkan tidak dimulai dengan ""). AWS

### Publikasikan data metrik kustom

Untuk mempublikasikan data metrik Anda sendiri, panggil `putMetricData` metode ini dengan file [PutMetricDataRequest](#). `CloudWatchClient` `PutMetricDataRequest` harus menyertakan namespace khusus yang akan digunakan untuk data, dan informasi tentang titik data itu sendiri dalam suatu [MetricDatum](#) objek.

#### Note

Anda tidak dapat menentukan namespace yang dimulai dengan ""AWS. Ruang nama yang dimulai dengan "AWS" dicadangkan untuk digunakan oleh Amazon Web Services produk.

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
```

## Kode

```
public static void putMetData(CloudWatchClient cw, Double dataPoint ) {

    try {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
            .value("URLS")
            .build();

        // Set an Instant object
        String time =
ZonedDateTime.now( ZoneOffset.UTC ).format( DateTimeFormatter.ISO_INSTANT );
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
            .value(dataPoint)
            .timestamp(instant)
            .dimensions(dimension).build();

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace("SITE/TRAFFIC")
            .metricData(datum).build();

        cw.putMetricData(request);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.printf("Successfully put data point %f", dataPoint);
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Informasi lain

- [Gunakan Amazon CloudWatch Metrik](#) di Panduan Amazon CloudWatch Pengguna.
- [AWS Ruang nama](#) di Amazon CloudWatch Panduan Pengguna.
- [PutMetricData](#) dalam Referensi Amazon CloudWatch API.



## Bekerja dengan CloudWatch alarm

### Membuat alarm

Untuk membuat alarm berdasarkan CloudWatch metrik, panggil `putMetricAlarm` metode dengan [PutMetricAlarmRequest](#) diisi dengan kondisi alarm. `CloudWatchClient`

#### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
```

#### Kode

```
public static void putMetricAlarm(CloudWatchClient cw, String alarmName, String
instanceId) {

    try {
        Dimension dimension = Dimension.builder()
            .name("InstanceId")
            .value(instanceId).build();

        PutMetricAlarmRequest request = PutMetricAlarmRequest.builder()
            .alarmName(alarmName)
            .comparisonOperator(
                ComparisonOperator.GREATER_THAN_THRESHOLD)
            .evaluationPeriods(1)
            .metricName("CPUUtilization")
            .namespace("AWS/EC2")
            .period(60)
            .statistic(Statistic.AVERAGE)
            .threshold(70.0)
            .actionsEnabled(false)
            .alarmDescription(
                "Alarm when server CPU utilization exceeds 70%")
            .unit(StandardUnit.SECONDS)
            .dimensions(dimension)
```

```
        .build();

        cw.putMetricAlarm(request);
        System.out.printf(
            "Successfully created alarm with name %s", alarmName);
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Daftar alarm

Untuk membuat daftar CloudWatch alarm yang telah Anda buat, panggil `describeAlarms` metode ini dengan [DescribeAlarmsRequest](#) yang dapat Anda gunakan untuk mengatur opsi untuk hasilnya.

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
```

## Kode

```
public static void desCWAAlarms( CloudWatchClient cw) {

    try {

        boolean done = false;
        String newToken = null;

        while(!done) {
            DescribeAlarmsResponse response;

            if (newToken == null) {
```

```

        DescribeAlarmsRequest request =
DescribeAlarmsRequest.builder().build();
        response = cw.describeAlarms(request);
    } else {
        DescribeAlarmsRequest request = DescribeAlarmsRequest.builder()
            .nextToken(newToken)
            .build();
        response = cw.describeAlarms(request);
    }

    for(MetricAlarm alarm : response.metricAlarms()) {
        System.out.printf("\n Retrieved alarm %s", alarm.alarmName());
    }

    if(response.nextToken() == null) {
        done = true;
    } else {
        newToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.printf("Done");
}

```

Daftar alarm dapat diperoleh dengan memanggil `MetricAlarms` [DescribeAlarmsResponse](#) yang dikembalikan oleh `describeAlarms`.

Hasilnya mungkin paged. Untuk mengambil kumpulan hasil berikutnya, panggil `nextToken` objek respons dan gunakan nilai token untuk membangun objek permintaan baru. Kemudian panggil `describeAlarms` metode lagi dengan permintaan baru.

#### Note

Anda juga dapat mengambil alarm untuk metrik tertentu dengan menggunakan metode ini `CloudWatchClient`. `describeAlarmsForMetric` Penggunaannya mirip dengan `describeAlarms`.

Lihat [contoh lengkapnya](#) di GitHub.

## Menghapus alarm

Untuk menghapus CloudWatch alarm, panggil `deleteAlarms` metode dengan [DeleteAlarmsRequest](#) berisi satu atau beberapa nama alarm yang ingin Anda hapus.

CloudWatchClient

Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
```

Kode

```
public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {

    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.printf("Successfully deleted alarm %s", alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Informasi lain

- [Menggunakan Amazon CloudWatch alarm](#) di Amazon CloudWatch Panduan Pengguna
- [PutMetricAlarm](#) di Referensi Amazon CloudWatch API
- [DescribeAlarms](#) di Referensi Amazon CloudWatch API

- [DeleteAlarms](#) di Referensi Amazon CloudWatch API

## Gunakan CloudWatch Acara Amazon

CloudWatch Acara memberikan aliran peristiwa sistem yang mendekati real-time yang menggambarkan perubahan AWS sumber daya ke Amazon EC2 instance, Lambda fungsi, Kinesis aliran, Amazon ECS tugas, mesin Step Functions status, Amazon SNS topik, Amazon SQS antrian, atau target bawaan. Anda dapat mencocokkan acara dan merutekannya ke satu atau beberapa fungsi atau aliran target dengan menggunakan aturan sederhana.

Amazon EventBridge adalah [evolusi](#) dari CloudWatch Peristiwa. Kedua layanan menggunakan API yang sama, sehingga Anda dapat terus menggunakan [klien CloudWatch Acara](#) yang disediakan oleh SDK atau bermigrasi ke fungsionalitas SDK untuk [EventBridge klien](#) Java untuk CloudWatch Acara. CloudWatch [Dokumentasi Panduan Pengguna](#) Acara dan [referensi API](#) sekarang tersedia melalui situs EventBridge dokumentasi.

### Tambahkan acara

Untuk menambahkan CloudWatch peristiwa khusus, panggil `CloudWatchEventsClient`'s `putEvents` metode dengan [PutEventsRequest](#) objek yang berisi satu atau beberapa [PutEventsRequestEntry](#) objek yang memberikan detail tentang setiap peristiwa. Anda dapat menentukan beberapa parameter untuk entri seperti sumber dan jenis acara, sumber daya yang terkait dengan acara, dan sebagainya.

#### Note

Anda dapat menentukan maksimum 10 acara per panggilan ke `putEvents`.

### Impor

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;
```

### Kode

```
public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn ) {
```

```
try {

    final String EVENT_DETAILS =
        "{ \"key1\": \"value1\", \"key2\": \"value2\" }";

    PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
        .detail(EVENT_DETAILS)
        .detailType("sampleSubmitted")
        .resources(resourceArn)
        .source("aws-sdk-java-cloudwatch-example")
        .build();

    PutEventsRequest request = PutEventsRequest.builder()
        .entries(requestEntry)
        .build();

    cwe.putEvents(request);
    System.out.println("Successfully put CloudWatch event");

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Tambahkan aturan

Untuk membuat atau memperbarui aturan, panggil `CloudWatchEventsClient`'s `putRule` metode [PutRuleRequest](#) dengan nama aturan dan parameter opsional seperti [pola acara](#), IAM peran untuk dikaitkan dengan aturan, dan [ekspresi penjadwalan](#) yang menjelaskan seberapa sering aturan dijalankan.

## Impor

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;
```

## Kode

```
public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
roleArn) {

    try {
        PutRuleRequest request = PutRuleRequest.builder()
            .name(ruleName)
            .roleArn(roleArn)
            .scheduleExpression("rate(5 minutes)")
            .state(RuleState.ENABLED)
            .build();

        PutRuleResponse response = cwe.putRule(request);
        System.out.printf(
            "Successfully created CloudWatch events rule %s with arn %s",
            roleArn, response.ruleArn());
    } catch (
        CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Tambahkan target

Target adalah sumber daya yang dipanggil ketika suatu aturan dipicu. Contoh target termasuk Amazon EC2 instance, Lambda fungsi, Kinesis aliran, Amazon ECS tugas, mesin Step Functions status, dan target bawaan.

Untuk menambahkan target ke aturan, panggil `CloudWatchEventsClient`'s `putTargets` metode dengan [PutTargetsRequest](#) berisi aturan untuk diperbarui dan daftar target untuk ditambahkan ke aturan.

## Impor

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsResponse;
```

```
import software.amazon.awssdk.services.cloudwatchevents.model.Target;
```

## Kode

```
public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName, String
functionArn, String targetId ) {

    try {
        Target target = Target.builder()
            .arn(functionArn)
            .id(targetId)
            .build();

        PutTargetsRequest request = PutTargetsRequest.builder()
            .targets(target)
            .rule(ruleName)
            .build();

        PutTargetsResponse response = cwe.putTargets(request);
        System.out.printf(
            "Successfully created CloudWatch events target for rule %s",
            ruleName);
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Informasi lain

- [Menambahkan Acara dengan PutEvents](#) di Panduan EventBridge Pengguna Amazon
- [Ekspresi Jadwal untuk Aturan](#) di Panduan EventBridge Pengguna Amazon
- [Jenis Acara untuk CloudWatch Events](#) di Panduan EventBridge Pengguna Amazon
- [Pola Acara](#) di Panduan EventBridge Pengguna Amazon
- [PutEvents](#) di Referensi EventBridge API Amazon
- [PutTargets](#) di Referensi EventBridge API Amazon
- [PutRule](#) di Referensi EventBridge API Amazon



## AWSlayanan database dan AWS SDK for Java 2.x

AWS [menawarkan beberapa jenis database: relasional, key-value, in-memory, document, dan beberapa lainnya](#). Dukungan SDK for Java 2.x bervariasi tergantung sifat layanan database di AWS.

Beberapa layanan database, misalnya layanan [Amazon DynamoDB](#), memiliki API layanan web untuk mengelola AWS sumber daya (database) serta API layanan web untuk berinteraksi dengan data.

[Dalam SDK for Java 2.x jenis layanan ini memiliki klien layanan khusus, misalnya DynamoDBClient.](#)

Layanan database lainnya memiliki API layanan web yang berinteraksi dengan sumber daya, seperti [Amazon DocumentDB](#) API (untuk cluster, instance dan manajemen sumber daya), tetapi tidak memiliki API layanan web untuk bekerja dengan data. SDK for Java 2.x memiliki antarmuka yang [DocDbClient](#) sesuai untuk bekerja dengan sumber daya. Namun, Anda memerlukan API Java lain, seperti [MongoDB untuk Java untuk](#) bekerja dengan data.

Gunakan contoh di bawah ini untuk mempelajari cara Anda menggunakan klien layanan SDK for Java 2.x dengan berbagai jenis database.

### Contoh Amazon DynamoDB

Bekerja dengan data

SDK service client: [DynamoDBClient](#)

Example: [Aplikasi React/Spring REST menggunakan DynamoDB](#)

Examples: [Beberapa contoh DynamoDB](#)

SDK service client: [DynamoDB EnhancedClient](#)

Example: [Aplikasi React/Spring REST menggunakan DynamoDB](#)

Examples: [Beberapa contoh DynamoDB](#) (names starting with 'Enhanced')

Bekerja dengan database

SDK service client: [DynamoDBClient](#)

Examples: [CreateTable, ListTables, DeleteTable](#)

Lihat contoh [DynamoDB tambahan](#) di bagian contoh kode terpandu dari panduan ini.

## Contoh-contoh Amazon RDS

Bekerja dengan data	Bekerja dengan database
API non-SDK: JDBC, ragam SQL khusus database; kode Anda mengelola koneksi database atau kumpulan koneksi.	Klien layanan SDK: <a href="#">RdsClient</a>
Contoh: Aplikasi <a href="#">React/Spring REST</a> menggunakan MySQL	Contoh: <a href="#">Beberapa RdsClient contoh</a>

## Contoh Amazon Redshift

Bekerja dengan data	Bekerja dengan database
Klien layanan SDK: <a href="#">RedshiftDataClient</a>	Klien layanan SDK: <a href="#">RedshiftClient</a>
Contoh: <a href="#">Beberapa RedshiftDataClient contoh</a>	Contoh: <a href="#">Beberapa RedshiftClient contoh</a>
Contoh: Aplikasi <a href="#">React/Spring</a> REST menggunakan RedshiftDataClient	

## Amazon Aurora Contoh v1 Tanpa Server

Bekerja dengan data	Bekerja dengan database
Klien layanan SDK: <a href="#">RdsDataClient</a>	Klien layanan SDK: <a href="#">RdsClient</a>
Contoh: Aplikasi <a href="#">React/Spring</a> REST menggunakan RdsDataClient	Contoh: <a href="#">Beberapa RdsClient contoh</a>

## Contoh Amazon DocumentDB

Bekerja dengan data	Bekerja dengan database
Non-SDK API: pustaka Java khusus MongoDB (misalnya <a href="#">MongoDB untuk Java</a> ); kode Anda mengelola koneksi database atau kumpulan koneksi.	Klien layanan SDK: <a href="#">DocDbClient</a>
Contoh: Panduan <a href="#">Pengembang DocumentDB (Mongo)</a> (pilih tab 'Java')	

## Bekerja dengan DynamoDB

Bagian ini memberikan contoh yang menunjukkan cara bekerja dengan [DynamoDB](#). Contoh di bagian ini menggunakan standar, tingkat rendah DynamoDB client [DynamoDbClient](#) () yang ditawarkan dengan 2.x. AWS SDK for Java

SDK juga menawarkan [DynamoDB Enhanced](#) Client yang menyediakan pendekatan berorientasi objek tingkat tinggi untuk bekerja dengan DynamoDB.

Topik

- [Bekerja dengan tabel di DynamoDB](#)
- [Bekerja dengan item di DynamoDB](#)
- [Peta objek Java ke item DynamoDB dengan AWS SDK for Java 2.x](#)

## Bekerja dengan tabel di DynamoDB

Tabel adalah wadah untuk semua item dalam DynamoDB database. Sebelum Anda dapat menambah atau menghapus data dari DynamoDB, Anda harus membuat tabel.

Untuk setiap tabel, Anda harus mendefinisikan:

- Nama tabel yang unik untuk akun dan Wilayah Anda.
- Kunci utama yang setiap nilainya harus unik; tidak ada dua item dalam tabel Anda yang dapat memiliki nilai kunci primer yang sama.

Kunci primer bisa sederhana, terdiri dari kunci partisi tunggal (HASH), atau komposit, yang terdiri dari partisi dan kunci sort (RANGE).

Setiap nilai kunci memiliki tipe data terkait, disebutkan oleh kelas. [ScalarAttributeType](#) Nilai kunci dapat berupa biner (B), numerik (N), atau string (S). Untuk informasi selengkapnya, lihat [Aturan Penamaan dan Jenis Data](#) di Panduan Amazon DynamoDB Pengembang.

- Throughput yang disediakan adalah nilai yang menentukan jumlah unit kapasitas baca/tulis yang dicadangkan untuk tabel.

#### Note

[Amazon DynamoDB Penetapan harga](#) didasarkan pada nilai throughput yang disediakan yang Anda tetapkan pada tabel Anda, jadi cadangkan hanya kapasitas sebanyak yang Anda pikir Anda perlukan untuk tabel Anda.

Throughput yang disediakan untuk tabel dapat dimodifikasi kapan saja, sehingga Anda dapat menyesuaikan kapasitas saat kebutuhan Anda berubah.

## Buat tabel

Gunakan `DynamoDbClient`'s `createTable` metode ini untuk membuat DynamoDB tabel baru. Anda perlu membangun atribut tabel dan skema tabel, yang keduanya digunakan untuk mengidentifikasi kunci utama tabel Anda. Anda juga harus menyediakan nilai throughput awal yang disediakan dan nama tabel.

#### Note

Jika tabel dengan nama yang Anda pilih sudah ada, sebuah [DynamoDbException](#) dilemparkan.

## Buat tabel dengan kunci primer sederhana

Kode ini membuat tabel dengan satu atribut yang merupakan kunci utama sederhana tabel. contoh menggunakan [AttributeDefinition](#) dan [KeySchemaElement](#) objek untuk.

### [CreateTableRequest](#)

## Impor

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

## Kode

```
public static String createTable(DynamoDbClient ddb, String tableName, String key)
{
    DynamoDbWaiter dbWaiter = ddb.waiter();
    CreateTableRequest request = CreateTableRequest.builder()
        .attributeDefinitions(AttributeDefinition.builder()
            .attributeName(key)
            .attributeType(ScalarAttributeType.S)
            .build())
        .keySchema(KeySchemaElement.builder()
            .attributeName(key)
            .keyType(KeyType.HASH)
            .build())
        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10))
            .build())
        .tableName(tableName)
        .build();

    String newTable = "";
    try {
        CreateTableResponse response = ddb.createTable(request);
    }
```

```

        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        // Wait until the Amazon DynamoDB table is created
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);

        newTable = response.tableDescription().tableName();
        return newTable;

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

```

Lihat [contoh lengkapnya](#) di GitHub.

Buat tabel dengan kunci primer komposit

Contoh berikut membuat tabel dengan dua atribut. Kedua atribut digunakan untuk kunci primer komposit.

Impor

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

```

Kode

```

public static String createTableComKey(DynamoDbClient ddb, String tableName) {

```

```
        CreateTableRequest request = CreateTableRequest.builder()
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("Language")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("Greeting")
                    .attributeType(ScalarAttributeType.S)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("Language")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("Greeting")
                    .keyType(KeyType.RANGE)
                    .build())
            .provisionedThroughput(
                ProvisionedThroughput.builder()
                    .readCapacityUnits(new Long(10))
                    .writeCapacityUnits(new Long(10)).build())
            .tableName(tableName)
            .build();

        String tableId = "";

        try {
            CreateTableResponse result = ddb.createTable(request);
            tableId = result.tableDescription().tableId();
            return tableId;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Mencantumkan tabel

Anda dapat membuat daftar tabel di Wilayah tertentu dengan memanggil `DynamoDbClient`'s `listTables` metode.

### Note

Jika tabel bernama tidak ada untuk akun dan Wilayah Anda, a [ResourceNotFoundException](#) dilemparkan.

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.List;
```

## Kode

```
public static void listAllTables(DynamoDbClient ddb){

    boolean moreTables = true;
    String lastName = null;

    while(moreTables) {
        try {
            ListTablesResponse response = null;
            if (lastName == null) {
                ListTablesRequest request = ListTablesRequest.builder().build();
                response = ddb.listTables(request);
            } else {
                ListTablesRequest request = ListTablesRequest.builder()
                    .exclusiveStartTableName(lastName).build();
                response = ddb.listTables(request);
            }

            List<String> tableNames = response.tableNames();

            if (tableNames.size() > 0) {
```



```
        for (String curName : tableNames) {
            System.out.format("* %s\n", curName);
        }
    } else {
        System.out.println("No tables found!");
        System.exit(0);
    }

    lastName = response.lastEvaluatedTableName();
    if (lastName == null) {
        moreTables = false;
    }
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
System.out.println("\nDone!");
}
```

Secara default, hingga 100 tabel dikembalikan per panggilan—gunakan `lastEvaluatedTableName` pada [ListTablesResponse](#) objek yang dikembalikan untuk mendapatkan tabel terakhir yang dievaluasi. Anda dapat menggunakan nilai ini untuk memulai daftar setelah nilai terakhir yang dikembalikan dari daftar sebelumnya.

Lihat [contoh lengkapnya](#) di GitHub.

## Jelaskan (dapatkan informasi tentang) tabel

Gunakan `DynamoDbClient`'s `describeTable` metode ini untuk mendapatkan informasi tentang tabel.

### Note

Jika tabel bernama tidak ada untuk akun dan Wilayah Anda, a [ResourceNotFoundException](#) dilemparkan.

## Impor

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;
```

## Kode

```
public static void describeDynamoDBTable(DynamoDbClient ddb, String tableName ) {

    DescribeTableRequest request = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        TableDescription tableInfo =
            ddb.describeTable(request).table();

        if (tableInfo != null) {
            System.out.format("Table name   : %s\n",
                tableInfo.tableName());
            System.out.format("Table ARN   : %s\n",
                tableInfo.tableArn());
            System.out.format("Status      : %s\n",
                tableInfo.tableStatus());
            System.out.format("Item count  : %d\n",
                tableInfo.itemCount().longValue());
            System.out.format("Size (bytes): %d\n",
                tableInfo.tableSizeBytes().longValue());

            ProvisionedThroughputDescription throughputInfo =
                tableInfo.provisionedThroughput();
            System.out.println("Throughput");
            System.out.format("  Read Capacity : %d\n",
                throughputInfo.readCapacityUnits().longValue());
            System.out.format("  Write Capacity: %d\n",
                throughputInfo.writeCapacityUnits().longValue());

            List<AttributeDefinition> attributes =
                tableInfo.attributeDefinitions();
            System.out.println("Attributes");
```

```
        for (AttributeDefinition a : attributes) {
            System.out.format("  %s (%s)\n",
                a.attributeName(), a.attributeType());
        }
    }
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("\nDone!");
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Ubah (perbarui) tabel

Anda dapat memodifikasi nilai throughput yang disediakan tabel kapan saja dengan memanggil metode `DynamoDbClient`'s `updateTable`

### Note

Jika tabel bernama tidak ada untuk akun dan Wilayah Anda, a [ResourceNotFoundException](#) dilemparkan.

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.UpdateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

## Kode

```
public static void updateDynamoDBTable(DynamoDbClient ddb,
                                       String tableName,
                                       Long readCapacity,
                                       Long writeCapacity) {

    System.out.format(
```

```
        "Updating %s with new provisioned throughput values\n",
        tableName);
System.out.format("Read capacity : %d\n", readCapacity);
System.out.format("Write capacity : %d\n", writeCapacity);

ProvisionedThroughput tableThroughput = ProvisionedThroughput.builder()
    .readCapacityUnits(readCapacity)
    .writeCapacityUnits(writeCapacity)
    .build();

UpdateTableRequest request = UpdateTableRequest.builder()
    .provisionedThroughput(tableThroughput)
    .tableName(tableName)
    .build();

try {
    ddb.updateTable(request);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

System.out.println("Done!");
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Hapus tabel

Untuk menghapus tabel, panggil `DynamoDbClient`'s `deleteTable` metode dan berikan nama tabel.

### Note

Jika tabel bernama tidak ada untuk akun dan Wilayah Anda, a [ResourceNotFoundException](#)dilemparkan.

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
```

## Kode

```
public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Informasi lain

- [Pedoman untuk Bekerja dengan Tabel](#) di Panduan Amazon DynamoDB Pengembang
- [Bekerja dengan Tabel DynamoDB di](#) Panduan Amazon DynamoDB Pengembang

## Bekerja dengan item di DynamoDB

Dalam DynamoDB, item adalah kumpulan atribut, yang masing-masing memiliki nama dan nilai. Nilai atribut dapat berupa skalar, set, atau jenis dokumen. Untuk informasi selengkapnya, lihat [Aturan Penamaan dan Jenis Data](#) di Panduan Amazon DynamoDB Pengembang.

### Ambil (dapatkan) item dari tabel

Panggil `getItem` metode ini dan berikan [GetItemRequest](#) objek dengan nama tabel dan nilai kunci primer dari item yang Anda inginkan. `DynamoDbClient` Ia mengembalikan [GetItemResponse](#) objek dengan semua atribut untuk item itu. Anda dapat menentukan satu atau lebih [ekspresi proyeksi](#) dalam `GetItemRequest` untuk mengambil atribut tertentu.

Anda dapat menggunakan `item()` metode `GetItemResponse` objek yang dikembalikan untuk mengambil [Map](#) of key (String) dan value ([AttributeValue](#)) pasangan yang terkait dengan item tersebut.

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
```

## Kode

```
public static void getDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal ) {

    HashMap<String, AttributeValue> keyToGet = new HashMap<String, AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal).build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName(tableName)
        .build();

    try {
        Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();

        if (returnedItem != null) {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");

            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
            }
        } else {
            System.out.format("No item found with the key %s!\n", key);
        }
    }
}
```

```
    }  
  } catch (DynamoDbException e) {  
    System.err.println(e.getMessage());  
    System.exit(1);  
  }  
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Ambil (dapatkan) item dari tabel menggunakan klien asinkron

Memanggil `getItem` metode `DynamoDbAsyncClient` dan meneruskannya [GetItemRequest](#) objek dengan nama tabel dan nilai kunci utama dari item yang Anda inginkan.

Anda dapat mengembalikan instance [Collection](#) dengan semua atribut untuk item tersebut (lihat contoh berikut).

### Impor

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;  
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;  
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Set;  
import java.util.stream.Collectors;  
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

### Kode

```
public static void getItem(DynamoDbAsyncClient client, String tableName, String  
key, String keyVal) {  
  
    HashMap<String, AttributeValue> keyToGet =  
        new HashMap<String, AttributeValue>();  
  
    keyToGet.put(key, AttributeValue.builder()  
        .s(keyVal).build());  
  
    try {
```

```
// Create a GetItemRequest instance
getItemRequest request = GetItemRequest.builder()
    .key(keyToGet)
    .tableName(tableName)
    .build();

// Invoke the DynamoDbAsyncClient object's getItem
java.util.Collection<AttributeValue> returnedItem =
client.getItem(request).join().item().values();

// Convert Set to Map
Map<String, AttributeValue> map =
returnedItem.stream().collect(Collectors.toMap(AttributeValue::s, s->s));
Set<String> keys = map.keySet();
for (String sinKey : keys) {
    System.out.format("%s: %s\n", sinKey, map.get(sinKey).toString());
}

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Tambahkan item baru ke tabel

Buat [Peta](#) pasangan kunci-nilai yang mewakili atribut item. Ini harus menyertakan nilai untuk bidang kunci utama tabel. Jika item yang diidentifikasi oleh kunci utama sudah ada, bidangnya diperbarui oleh permintaan.

### Note

Jika tabel bernama tidak ada untuk akun dan wilayah Anda, a [ResourceNotFoundException](#) dilemparkan.

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
```



```
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;
```

## Kode

```
public static void putItemInTable(DynamoDbClient ddb,
                                  String tableName,
                                  String key,
                                  String keyVal,
                                  String albumTitle,
                                  String albumTitleValue,
                                  String awards,
                                  String awardVal,
                                  String songTitle,
                                  String songTitleVal){

    HashMap<String,AttributeValue> itemValues = new
    HashMap<String,AttributeValue>();

    // Add all content to the table
    itemValues.put(key, AttributeValue.builder().s(keyVal).build());
    itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
    itemValues.put(albumTitle,
    AttributeValue.builder().s(albumTitleValue).build());
    itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

    PutItemRequest request = PutItemRequest.builder()
        .tableName(tableName)
        .item(itemValues)
        .build();

    try {
        ddb.putItem(request);
        System.out.println(tableName + " was successfully updated");

    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be found.
        \n", tableName);
        System.err.println("Be sure that it exists and that you've typed its name
        correctly!");
        System.exit(1);
    }
}
```

```
    } catch (DynamoDbException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Memperbarui item yang ada dalam tabel

Anda dapat memperbarui atribut untuk item yang sudah ada dalam tabel dengan menggunakan `updateItem` metode, memberikan nama tabel, nilai kunci primer, dan peta bidang yang akan diperbarui. `DynamoDbClient`

### Note

Jika tabel bernama tidak ada untuk akun dan wilayah Anda, atau jika item yang diidentifikasi oleh kunci utama yang Anda lewati tidak ada, akan [ResourceNotFoundException](#) ditampilkan.

## Impor

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;  
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;  
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;  
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;  
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;  
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;  
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
import java.util.HashMap;
```

## Kode

```
public static void updateTableItem(DynamoDbClient ddb,  
                                   String tableName,  
                                   String key,  
                                   String keyVal,  
                                   String name,
```

```
String updateVal){

    HashMap<String,AttributeValue> itemKey = new HashMap<String,AttributeValue>();

    itemKey.put(key, AttributeValue.builder().s(keyVal).build());

    HashMap<String,AttributeValueUpdate> updatedValues =
        new HashMap<String,AttributeValueUpdate>();

    // Update the column specified by name with updatedVal
    updatedValues.put(name, AttributeValueUpdate.builder()
        .value(AttributeValue.builder().s(updateVal).build())
        .action(AttributeAction.PUT)
        .build());

    UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(itemKey)
        .attributeUpdates(updatedValues)
        .build();

    try {
        ddb.updateItem(request);
    } catch (ResourceNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    System.out.println("Done!");
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Menghapus item yang ada dalam tabel

Anda dapat menghapus item yang ada dalam tabel dengan menggunakan `deleteItem` metode dan memberikan nama tabel serta nilai kunci primer. `DynamoDbClient`

**Note**

Jika tabel bernama tidak ada untuk akun dan wilayah Anda, atau jika item yang diidentifikasi oleh kunci utama yang Anda lewati tidak ada, akan [ResourceNotFoundException](#) ditampilkan.

**Impor**

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;
```

**Kode**

```
public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal) {

    HashMap<String,AttributeValue> keyToGet =
        new HashMap<String,AttributeValue>();

    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal)
        .build());

    DeleteItemRequest deleteReq = DeleteItemRequest.builder()
        .tableName(tableName)
        .key(keyToGet)
        .build();

    try {
        ddb.deleteItem(deleteReq);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Informasi lain

- [Pedoman untuk Bekerja dengan Item](#) di Panduan Amazon DynamoDB Pengembang
- [Bekerja dengan Item DynamoDB di](#) Panduan Amazon DynamoDB Pengembang

## Peta objek Java ke item DynamoDB dengan AWS SDK for Java 2.x

[DynamoDB Enhanced Client](#) API adalah library tingkat tinggi yang merupakan penerus kelas dalam SDK for DynamoDBMapper Java v1.x. Ini menawarkan cara mudah untuk memetakan kelas sisi klien ke tabel DynamoDB. Anda menentukan hubungan antara tabel dan kelas model yang sesuai dalam kode Anda. Setelah Anda menentukan hubungan tersebut, Anda dapat secara intuitif melakukan berbagai operasi membuat, membaca, memperbarui, atau menghapus (CRUD) pada tabel atau item di DynamoDB.

DynamoDB Enhanced Client API juga menyertakan API [Dokumen yang Ditingkatkan](#) yang memungkinkan Anda bekerja dengan item tipe dokumen yang tidak mengikuti skema yang ditentukan.

DynamoDB Enhanced Client API dibahas dalam topik berikut.

- [Memulai menggunakan DynamoDB Enhanced Client API](#)
- [Pelajari dasar-dasar DynamoDB Enhanced Client API](#)
- [Gunakan fitur pemetaan tingkat lanjut](#)
- [Bekerja dengan dokumen JSON dengan Enhanced Document API untuk DynamoDB](#)
- [Gunakan ekstensi](#)
- [Gunakan DynamoDB Enhanced Client API secara asinkron](#)
- [Anotasi kelas data](#)

## Memulai menggunakan DynamoDB Enhanced Client API

Tutorial berikut memperkenalkan Anda pada dasar-dasar yang Anda butuhkan untuk bekerja dengan DynamoDB Enhanced Client API.

## Tambahkan dependensi

Untuk mulai bekerja dengan DynamoDB Enhanced Client API di project Anda, tambahkan dependensi pada artefak Maven. dynamodb-enhanced ini ditunjukkan dalam contoh berikut.

### Maven

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

<VERSION> Lakukan pencarian repositori pusat Maven untuk [versi terbaru](#) dan ganti dengan nilai ini.

### Gradle

```
repositories {
    mavenCentral()
}
dependencies {
    implementation(platform("software.amazon.awssdk:bom:<VERSION>"))
    implementation("software.amazon.awssdk:dynamodb-enhanced")
    ...
}
```

<VERSION>Lakukan pencarian repositori pusat Maven untuk [versi terbaru](#) dan ganti dengan nilai ini.

Menghasilkan **TableSchema** dari kelas data

A [TableSchema](#) memungkinkan klien yang disempurnakan untuk memetakan nilai atribut DynamoDB ke dan dari kelas sisi klien Anda. Dalam tutorial ini, Anda belajar tentang `TableSchema` yang berasal dari kelas data statis dan dihasilkan dari kode dengan menggunakan pembangun.

Gunakan kelas data berannotasi

SDK for Java 2.x menyertakan [serangkaian anotasi](#) yang dapat Anda gunakan dengan kelas data untuk menghasilkan `TableSchema` cepat untuk memetakan kelas Anda ke tabel.

Mulailah dengan membuat kelas data yang sesuai dengan [JavaBean spesifikasi](#). Spesifikasi mensyaratkan bahwa kelas memiliki konstruktor publik tanpa argumen dan memiliki getter dan setter untuk setiap atribut di kelas. Sertakan anotasi tingkat kelas untuk menunjukkan bahwa kelas data adalah a. `DynamoDbBean` Juga, minimal, sertakan `DynamoDbPartitionKey` anotasi pada pengambil atau penyetel untuk atribut kunci utama.

Anda dapat menerapkan [anotasi tingkat atribut](#) ke getter atau setter, tetapi tidak keduanya.

#### Note

Istilah `property` ini biasanya digunakan untuk nilai yang dikapsulasi dalam a. `JavaBean` Namun, panduan ini menggunakan istilah `attribute` sebagai gantinya, agar konsisten dengan terminologi yang digunakan oleh DynamoDB.

`Customer` kelas berikut menunjukkan anotasi yang menghubungkan definisi kelas ke tabel DynamoDB.

**Customer** kelas

```
package org.example.tests.model;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
```

```
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.time.Instant;

@DynamoDbBean
public class Customer {

    private String id;
    private String name;
    private String email;
    private Instant regDate;

    @DynamoDbPartitionKey
    public String getId() { return this.id; }

    public void setId(String id) { this.id = id; }

    public String getCustName() { return this.name; }

    public void setCustName(String name) { this.name = name; }

    @DynamoDbSortKey
    public String getEmail() { return this.email; }

    public void setEmail(String email) { this.email = email; }

    public Instant getRegistrationDate() { return this.regDate; }

    public void setRegistrationDate(Instant registrationDate) { this.regDate =
registrationDate; }

    @Override
    public String toString() {
        return "Customer [id=" + id + ", name=" + name + ", email=" + email
            + ", regDate=" + regDate + "];"
    }
}
```

Setelah Anda membuat kelas data berannotasi, gunakan untuk membuat `TableSchema`, seperti yang ditunjukkan pada cuplikan berikut.

```
static final TableSchema<Customer> customerTableSchema =
    TableSchema.fromBean(Customer.class);
```



A `TableSchema` dirancang untuk menjadi statis dan tidak dapat diubah. Anda biasanya dapat membuat instance pada waktu pemuatan kelas.

Metode `TableSchema.fromBean()` pabrik statis mengintrospeksi kacang untuk menghasilkan pemetaan atribut kelas data ke dan dari atribut DynamoDB.

Untuk contoh bekerja dengan model data yang terdiri dari beberapa kelas data, lihat `Person` kelas di [???](#) bagian tersebut.

Gunakan pembangun

Anda dapat melewati biaya introspeksi kacang jika Anda menentukan skema tabel dalam kode. Jika Anda membuat kode skema, kelas Anda tidak perlu mengikuti standar JavaBean penamaan dan juga tidak perlu dianotasi. Contoh berikut menggunakan pembangun dan setara dengan contoh `Customer` kelas yang menggunakan anotasi.

```
static final TableSchema<Customer> customerTableSchema =
    TableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("id")
            .getter(Customer::getId)
            .setter(Customer::setId)
            .tags(StaticAttributeTags.primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("email")
            .getter(Customer::getEmail)
            .setter(Customer::setEmail)
            .tags(StaticAttributeTags.primarySortKey()))
        .addAttribute(String.class, a -> a.name("name")
            .getter(Customer::getCustName)
            .setter(Customer::setCustName))
        .addAttribute(Instant.class, a -> a.name("registrationDate")
            .getter(Customer::getRegistrationDate)
            .setter(Customer::setRegistrationDate))
        .build();
```

Buat klien yang disempurnakan dan **DynamoDbTable**

Buat klien yang disempurnakan

[DynamoDbEnhancedClient](#) kelas atau rekan asinkron, [DynamoDbEnhancedAsyncClient](#), adalah titik masuk untuk bekerja dengan DynamoDB Enhanced Client API.

Klien yang disempurnakan membutuhkan standar [DynamoDbClient](#) untuk melakukan pekerjaan. API menawarkan dua cara untuk membuat `DynamoDbEnhancedClient` instance. Opsi pertama, yang ditunjukkan dalam cuplikan berikut, membuat standar `DynamoDbClient` dengan pengaturan default diambil dari pengaturan konfigurasi.

```
DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.create();
```

Jika Anda ingin mengonfigurasi klien standar yang mendasarinya, Anda dapat menyediakannya ke metode pembangun klien yang disempurnakan seperti yang ditunjukkan pada cuplikan berikut.

```
DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
    .dynamoDbClient(
        // Configure an instance of the standard client.
        DynamoDbClient.builder()
            .region(Region.US_EAST_1)

        .credentialsProvider(ProfileCredentialsProvider.create())
            .build())
    .build();
```

Buat sebuah **DynamoDbTable** instance

Pikirkan a [DynamoDbTable](#) sebagai representasi sisi klien dari tabel DynamoDB yang menggunakan fungsionalitas pemetaan yang disediakan oleh file. `TableSchema` `DynamoDbTableKelas` menyediakan metode untuk operasi CRUD yang memungkinkan Anda berinteraksi dengan tabel DynamoDB tunggal.

`DynamoDbTable<T>` adalah kelas generik yang mengambil argumen tipe tunggal, apakah itu kelas khusus atau `EnhancedDocument` saat bekerja dengan item tipe dokumen. Jenis argumen ini menetapkan hubungan antara kelas yang Anda gunakan dan tabel DynamoDB tunggal.

Gunakan metode `table()` pabrik `DynamoDbEnhancedClient` untuk membuat `DynamoDbTable` instance seperti yang ditunjukkan pada cuplikan berikut.

```
static final DynamoDbTable<Customer> customerTable =
    enhancedClient.table("Customer", TableSchema.fromBean(Customer.class));
```

`DynamoDbTable` contoh adalah kandidat untuk lajang karena mereka tidak dapat diubah dan dapat digunakan di seluruh aplikasi Anda.

Kode Anda sekarang memiliki representasi dalam memori dari tabel DynamoDB yang dapat menyimpan instance. `Customer` Tabel DynamoDB yang sebenarnya mungkin atau mungkin tidak ada. Jika tabel bernama `Customer` sudah ada, Anda dapat mulai melakukan operasi CRUD terhadapnya. Jika tidak ada, gunakan `DynamoDbTable` instance untuk membuat tabel seperti yang dibahas di bagian berikutnya.

Buat tabel DynamoDB jika diperlukan

Setelah Anda membuat `DynamoDbTable` instance, gunakan untuk melakukan pembuatan tabel satu kali di DynamoDB.

Buat kode contoh tabel

Contoh berikut membuat tabel DynamoDB berdasarkan `Customer` kelas data.

Contoh ini membuat tabel DynamoDB dengan `Customer` nama —identik dengan nama kelas—tetapi nama tabel bisa menjadi sesuatu yang lain. Apa pun nama Anda tabel, Anda harus menggunakan nama ini dalam aplikasi tambahan untuk bekerja dengan tabel. Berikan nama ini ke `table()` metode kapan pun Anda membuat `DynamoDbTable` objek lain agar dapat bekerja dengan tabel DynamoDB yang mendasarinya.

Parameter lambda `Javabuilder`, diteruskan ke `createTable` metode memungkinkan Anda [menyesuaikan tabel](#). Dalam contoh ini, [throughput yang disediakan dikonfigurasi](#). Jika Anda ingin menggunakan pengaturan default saat membuat tabel, lewati pembangun seperti yang ditunjukkan pada cuplikan berikut.

```
customerDynamoDbTable.createTable();
```

Saat pengaturan default digunakan, nilai untuk throughput yang disediakan tidak disetel. Sebagai gantinya, mode penagihan untuk tabel diatur ke [sesuai permintaan](#).

Contoh ini juga menggunakan [DynamoDbWaiter](#) sebelum mencoba untuk mencetak nama tabel yang diterima dalam respon. Pembuatan meja membutuhkan waktu. Oleh karena itu, menggunakan pelayan berarti Anda tidak perlu menulis logika yang melakukan polling layanan DynamoDB untuk melihat apakah tabel ada sebelum menggunakan tabel.

Impor

```
import com.example.dynamodb.Customer;
```

```
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

## Kode

```
public static void createCustomerTable(DynamoDbTable<Customer> customerDynamoDbTable,
DynamoDbClient dynamoDbClient) {
    // Create the DynamoDB table by using the 'customerDynamoDbTable' DynamoDbTable
instance.
    customerDynamoDbTable.createTable(builder -> builder
        .provisionedThroughput(b -> b
            .readCapacityUnits(10L)
            .writeCapacityUnits(10L)
            .build())
    );
    // The 'dynamoDbClient' instance that's passed to the builder for the
DynamoDbWaiter is the same instance
    // that was passed to the builder of the DynamoDbEnhancedClient instance used to
create the 'customerDynamoDbTable'.
    // This means that the same Region that was configured on the standard
'dynamoDbClient' instance is used for all service clients.
    try (DynamoDbWaiter waiter =
DynamoDbWaiter.builder().client(dynamoDbClient).build()) { // DynamoDbWaiter is
Autocloseable
        ResponseOrException<DescribeTableResponse> response = waiter
            .waitUntilTableExists(builder -> builder.tableName("Customer").build())
            .matched();
        DescribeTableResponse tableDescription = response.response().orElseThrow(
            () -> new RuntimeException("Customer table was not created."));
        // The actual error can be inspected in response.exception()
        logger.info("Customer table was created.");
    }
}
```

### Note

Nama atribut tabel DynamoDB dimulai dengan huruf kecil ketika tabel dihasilkan dari kelas data. Jika Anda ingin nama atribut tabel dimulai dengan huruf besar, gunakan

`@DynamoDbAttribute(NAME)` anotasi dan berikan nama yang Anda inginkan sebagai parameter.

Lakukan operasi

Setelah tabel dibuat, gunakan `DynamoDbTable` instance untuk melakukan operasi terhadap tabel DynamoDB.

Dalam contoh berikut, singleton `DynamoDbTable<Customer>` dilewatkan sebagai parameter bersama dengan contoh [kelas Customer data](#) untuk menambahkan item baru ke tabel.

```
public static void putItemExample(DynamoDbTable<Customer> customerTable, Customer
customer){
    logger.info(customer.toString());
    customerTable.putItem(customer);
}
```

Objek **Customer**

```
Customer customer = new Customer();
customer.setId("1");
customer.setCustName("Customer Name");
customer.setEmail("customer@example.com");
customer.setRegistrationDate(Instant.parse("2023-07-03T10:15:30.00Z"));
```

Sebelum mengirim `customer` objek ke layanan DynamoDB, catat output dari metode objek untuk membandingkannya dengan apa yang dikirim `toString()` klien yang disempurnakan.

```
Customer [id=1, name=Customer Name, email=customer@example.com,
regDate=2023-07-03T10:15:30Z]
```

Pencatatan tingkat kabel menunjukkan muatan permintaan yang dihasilkan. Klien yang disempurnakan menghasilkan representasi tingkat rendah dari kelas data. `regDateAtribut`, yang merupakan `Instant` tipe di Java, direpresentasikan sebagai string DynamoDB.

```
{
  "TableName": "Customer",
  "Item": {
```

```
"registrationDate": {
  "S": "2023-07-03T10:15:30Z"
},
"id": {
  "S": "1"
},
"custName": {
  "S": "Customer Name"
},
"email": {
  "S": "customer@example.com"
}
}
}
```

## Bekerja dengan tabel yang ada

Bagian sebelumnya menunjukkan cara membuat tabel DynamoDB dimulai dengan kelas data Java. Jika Anda sudah memiliki tabel yang ada dan ingin menggunakan fitur klien yang disempurnakan, Anda dapat membuat kelas data Java untuk bekerja dengan tabel. Anda perlu memeriksa tabel DynamoDB dan menambahkan anotasi yang diperlukan ke kelas data.

Sebelum Anda bekerja dengan tabel yang ada, panggil `DynamoDbEnhanced.table()` metode. Ini dilakukan pada contoh sebelumnya dengan pernyataan berikut.

```
DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer",
    TableSchema.fromBean(Customer.class));
```

Setelah `DynamoDbTable` instance dikembalikan, Anda dapat mulai bekerja segera dengan tabel yang mendasarinya. Anda tidak perlu membuat ulang tabel dengan memanggil `DynamoDbTable.createTable()` metode.

Contoh berikut menunjukkan ini dengan segera mengambil `Customer` contoh dari tabel DynamoDB.

```
DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer",
    TableSchema.fromBean(Customer.class));
// The Customer table exists already and has an item with a primary key value of "1"
// and a sort key value of "customer@example.com".
customerTable.getItem(
    Key.builder().
        partitionValue("1").
```

```
sortValue("customer@example.com").build());
```

### ⚠ Important

Nama tabel yang digunakan dalam `table()` metode harus cocok dengan nama tabel DynamoDB yang ada.

## Pelajari dasar-dasar DynamoDB Enhanced Client API

[Topik ini membahas fitur dasar DynamoDB Enhanced Client API dan membandingkannya dengan API klien DynamoDB standar.](#)

Jika Anda baru mengenal DynamoDB Enhanced Client API, kami sarankan Anda melalui tutorial [pengantar](#) untuk membiasakan diri dengan kelas fundamental.

item DynamoDB di Java

Tabel DynamoDB menyimpan item. Bergantung pada kasus penggunaan Anda, item di sisi Java dapat berbentuk data atau struktur terstruktur statis yang dibuat secara dinamis.

Jika kasus penggunaan Anda memanggil item dengan kumpulan atribut yang konsisten, gunakan [kelas berannotasi](#) atau gunakan [pembuat](#) untuk menghasilkan tipe statis yang sesuai. `TableSchema`

Atau, jika Anda perlu menyimpan item yang terdiri dari berbagai struktur, buat `DocumentTableSchema`. `DocumentTableSchema` adalah bagian dari [Enhanced Document API](#) dan hanya memerlukan kunci primer yang diketik secara statis dan bekerja dengan `EnhancedDocument` instance untuk menyimpan elemen data. Enhanced Document API dibahas dalam [topik lain](#).

Jenis atribut

Meskipun DynamoDB [mendukung sejumlah kecil tipe atribut](#) dibandingkan dengan sistem tipe kaya Java, DynamoDB Enhanced Client API menyediakan mekanisme untuk mengonversi anggota kelas Java ke dan dari tipe atribut DynamoDB.

[Secara default, DynamoDB Enhanced Client API mendukung konverter atribut untuk sejumlah besar tipe, seperti Integer, String, dan Instant. `BigDecimal` Daftar ini muncul di kelas implementasi `AttributeConverter` antarmuka yang dikenal.](#) Daftar ini mencakup banyak jenis dan koleksi seperti peta, daftar, dan set.

Untuk menyimpan data untuk jenis atribut yang tidak didukung secara default atau tidak sesuai dengan JavaBean konvensi, Anda dapat menulis `AttributeConverter` implementasi kustom untuk melakukan konversi. Lihat bagian konversi atribut untuk [contoh](#).

Untuk menyimpan data untuk tipe atribut yang kelasnya sesuai dengan spesifikasi kacang Java (atau [kelas data yang tidak dapat diubah](#)), Anda dapat mengambil dua pendekatan.

- Jika Anda memiliki akses ke file sumber, Anda dapat membuat anotasi kelas dengan `@DynamoDbBean` (atau `@DynamoDbImmutable`). Bagian yang membahas atribut bersarang menunjukkan [contoh](#) penggunaan kelas beranotasi.
- Jika tidak memiliki akses ke file sumber dari kelas JavaBean data untuk atribut (atau Anda tidak ingin membubuhi keterangan file sumber kelas yang Anda memiliki akses ke), maka Anda dapat menggunakan pendekatan builder. Ini menciptakan skema tabel tanpa mendefinisikan kunci. Kemudian, Anda dapat menyarangkan skema tabel ini di dalam skema tabel lain untuk melakukan pemetaan. Bagian atribut bersarang memiliki [contoh](#) yang menunjukkan penggunaan skema bersarang.

## Nilai tipe primitif Java

Meskipun klien yang disempurnakan dapat bekerja dengan atribut tipe primitif, kami mendorong penggunaan tipe objek karena Anda tidak dapat mewakili nilai nol dengan tipe primitif.

## Nilai nol

Saat Anda menggunakan `putItem` API, klien yang disempurnakan tidak menyertakan atribut bernilai nol dari objek data yang dipetakan dalam permintaan ke DynamoDB.

Untuk `updateItem` permintaan, atribut bernilai nol dihapus dari item pada database. Jika Anda bermaksud memperbarui beberapa nilai atribut dan menjaga yang lain tidak berubah, salin nilai atribut lain yang tidak boleh diubah atau gunakan metode [ignoreNull\(\)](#) pada pembuat pembaruan.

Contoh berikut menunjukkan `ignoreNulls()` the `updateItem()` metode.

```
public void updateItemNullsExample(){
    Customer customer = new Customer();
    customer.setCustName("CustName");
    customer.setEmail("email");
    customer.setId("1");
}
```



```

    customer.setRegistrationDate(Instant.now());

    // Put item with values for all attributes.
    customerDynamoDbTable.putItem(customer);

    // Create a Customer instance with the same id value, but a different name
value.
    // Do not set the 'registrationDate' attribute.
    Customer custForUpdate = new Customer();
    custForUpdate.setCustName("NewName");
    custForUpdate.setEmail("email");
    custForUpdate.setId("1");

    // Update item without setting the registrationDate attribute.
    customerDynamoDbTable.updateItem(b -> b
        .item(custForUpdate)
        .ignoreNulls(Boolean.TRUE));

    Customer updatedWithNullsIgnored = customerDynamoDbTable.getItem(customer);
    // registrationDate value is unchanged.
    logger.info(updatedWithNullsIgnored.toString());

    customerDynamoDbTable.updateItem(custForUpdate);
    Customer updatedWithNulls = customerDynamoDbTable.getItem(customer);
    // registrationDate value is null because ignoreNulls() was not used.
    logger.info(updatedWithNulls.toString());
}
}

// Logged lines.
Customer [id=1, custName=NewName, email=email,
registrationDate=2023-04-05T16:32:32.056Z]
Customer [id=1, custName=NewName, email=email, registrationDate=null]

```

## Metode dasar DynamoDB Enhanced Client

Metode dasar peta klien yang disempurnakan ke operasi layanan DynamoDB yang dinamai menurut namanya. Contoh berikut menunjukkan variasi paling sederhana dari setiap metode. Anda dapat menyesuaikan setiap metode dengan meneruskan objek permintaan yang disempurnakan. Objek permintaan yang disempurnakan menawarkan sebagian besar fitur yang tersedia di klien DynamoDB standar. Mereka sepenuhnya didokumentasikan dalam Referensi AWS SDK for Java 2.x API.

Contoh menggunakan yang [the section called “Customer kelas”](#) ditunjukkan sebelumnya.

```
// CreateTable
customerTable.createTable();

// GetItem
Customer customer =
    customerTable.getItem(Key.builder().partitionValue("a123").build());

// UpdateItem
Customer updatedCustomer = customerTable.updateItem(customer);

// PutItem
customerTable.putItem(customer);

// DeleteItem
Customer deletedCustomer =
    customerTable.deleteItem(Key.builder().partitionValue("a123").sortValue(456).build());

// Query
PageIterable<Customer> customers = customerTable.query(keyEqualTo(k ->
    k.partitionValue("a123")));

// Scan
PageIterable<Customer> customers = customerTable.scan();

// BatchGetItem
BatchGetResultPageIterable batchResults =
    enhancedClient.batchGetItem(r -> r.addReadBatch(ReadBatch.builder(Customer.class)
        .mappedTableResource(customerTable)
        .addItem(key1)
        .addItem(key2)
        .addItem(key3)
        .build()));

// BatchWriteItem
batchResults = enhancedClient.batchWriteItem(r ->
    r.addWriteBatch(WriteBatch.builder(Customer.class)
        .mappedTableResource(customerTable)
        .addItem(customer)
        .deleteItem(key1)
        .deleteItem(key1)
        .build()));

// TransactGetItems
```

```

transactResults = enhancedClient.transactGetItems(r -> r.addItem(customerTable,
    key1)
                                                .addItem(customerTable,
    key2));

// TransactWriteItems
enhancedClient.transactWriteItems(r -> r.addConditionCheck(customerTable,
    i -> i.key(orderKey)
    .conditionExpression(conditionExpression))
    .addItem(customerTable, customer)
    .deleteItem(customerTable, key));

```

Bandingkan DynamoDB Enhanced Client dengan klien DynamoDB standar

Baik API klien DynamoDB — standar dan yang disempurnakan —memungkinkan Anda bekerja dengan tabel DynamoDB untuk melakukan operasi tingkat data CRUD (membuat, membaca, memperbarui, dan menghapus). Perbedaan antara API klien adalah bagaimana hal itu dicapai. Menggunakan klien standar, Anda bekerja secara langsung dengan atribut data tingkat rendah. API klien yang disempurnakan menggunakan kelas Java yang sudah dikenal dan memetakan ke API tingkat rendah di belakang layar.

Sementara kedua API klien mendukung operasi tingkat data, klien DynamoDB standar juga mendukung operasi tingkat sumber daya. Operasi tingkat sumber daya mengelola database, seperti membuat cadangan, daftar tabel, dan memperbarui tabel. API klien yang disempurnakan mendukung sejumlah operasi tingkat sumber daya tertentu seperti membuat, mendeskripsikan, dan menghapus tabel.

Untuk mengilustrasikan pendekatan berbeda yang digunakan oleh dua API klien, contoh kode berikut menunjukkan pembuatan ProductCatalog tabel yang sama menggunakan klien standar dan klien yang disempurnakan.

Bandingkan: Buat tabel menggunakan klien DynamoDB standar

```

DependencyFactory.dynamoDbClient().createTable(builder -> builder
    .tableName(TABLE_NAME)
    .attributeDefinitions(
        b -> b.attributeName("id").attributeType(ScalarAttributeType.N),
        b -> b.attributeName("title").attributeType(ScalarAttributeType.S),
        b -> b.attributeName("isbn").attributeType(ScalarAttributeType.S)
    )
    .keySchema(

```

```

        builder1 -> builder1.attributeName("id").keyType(KeyType.HASH),
        builder2 -> builder2.attributeName("title").keyType(KeyType.RANGE)
    )
    .globalSecondaryIndexes(builder3 -> builder3
        .indexName("products_by_isbn")
        .keySchema(builder2 -> builder2
            .attributeName("isbn").keyType(KeyType.HASH))
        .projection(builder2 -> builder2
            .projectionType(ProjectionType.INCLUDE)
            .nonKeyAttributes("price", "authors"))
        .provisionedThroughput(builder4 -> builder4
            .writeCapacityUnits(5L).readCapacityUnits(5L))
    )
    .provisionedThroughput(builder1 -> builder1
        .readCapacityUnits(5L).writeCapacityUnits(5L))
);

```

## Bandingkan: Buat tabel menggunakan DynamoDB Enhanced Client

```

DynamoDbEnhancedClient enhancedClient = DependencyFactory.enhancedClient();
productCatalog = enhancedClient.table(TABLE_NAME,
    TableSchema.fromImmutableClass(ProductCatalog.class));
productCatalog.createTable(b -> b
    .provisionedThroughput(b1 -> b1.readCapacityUnits(5L).writeCapacityUnits(5L))
    .globalSecondaryIndices(b2 -> b2.indexName("products_by_isbn")
        .projection(b4 -> b4
            .projectionType(ProjectionType.INCLUDE)
            .nonKeyAttributes("price", "authors"))
        .provisionedThroughput(b3 ->
            b3.writeCapacityUnits(5L).readCapacityUnits(5L))
    )
);

```

Klien yang disempurnakan menggunakan kelas data beranotasi berikut. DynamoDB Enhanced Client memetakan tipe data Java ke tipe data DynamoDB untuk kode verbose yang lebih mudah diikuti. `ProductCatalog` adalah contoh menggunakan kelas yang tidak dapat diubah dengan DynamoDB Enhanced Client. Penggunaan kelas `Immutable` untuk kelas data yang dipetakan [dibahas nanti dalam topik ini](#).

### **ProductCatalog** kelas

```
package org.example.tests.model;
```

```
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbIgnore;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbImmutable;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.math.BigDecimal;
import java.util.Objects;
import java.util.Set;

@DynamoDbImmutable(builder = ProductCatalog.Builder.class)
public class ProductCatalog implements Comparable<ProductCatalog> {
    private Integer id;
    private String title;
    private String isbn;
    private Set<String> authors;
    private BigDecimal price;

    private ProductCatalog(Builder builder){
        this.authors = builder.authors;
        this.id = builder.id;
        this.isbn = builder.isbn;
        this.price = builder.price;
        this.title = builder.title;
    }

    public static Builder builder(){ return new Builder(); }

    @DynamoDbPartitionKey
    public Integer id() { return id; }

    @DynamoDbSortKey
    public String title() { return title; }

    @DynamoDbSecondaryPartitionKey(indexNames = "products_by_isbn")
    public String isbn() { return isbn; }
    public Set<String> authors() { return authors; }
    public BigDecimal price() { return price; }
```

```
public static final class Builder {
    private Integer id;
    private String title;
    private String isbn;
    private Set<String> authors;
    private BigDecimal price;
    private Builder(){

    }

    public Builder id(Integer id) { this.id = id; return this; }
    public Builder title(String title) { this.title = title; return this; }
    public Builder isbn(String ISBN) { this.isbn = ISBN; return this; }
    public Builder authors(Set<String> authors) { this.authors = authors; return
this; }
    public Builder price(BigDecimal price) { this.price = price; return this; }
    public ProductCatalog build() { return new ProductCatalog(this); }
}

@Override
public String toString() {
    final StringBuffer sb = new StringBuffer("ProductCatalog{");
    sb.append("id=").append(id);
    sb.append(", title=").append(title).append('\ ');
    sb.append(", isbn=").append(isbn).append('\ ');
    sb.append(", authors=").append(authors);
    sb.append(", price=").append(price);
    sb.append('}');
    return sb.toString();
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    ProductCatalog that = (ProductCatalog) o;
    return id.equals(that.id) && title.equals(that.title) && Objects.equals(isbn,
that.isbn) && Objects.equals(authors, that.authors) && Objects.equals(price,
that.price);
}

@Override
public int hashCode() {
    return Objects.hash(id, title, isbn, authors, price);
}
```

```

@Override
@DynamoDbIgnore
public int compareTo(ProductCatalog other) {
    if (this.id.compareTo(other.id) != 0){
        return this.id.compareTo(other.id);
    } else {
        return this.title.compareTo(other.title);
    }
}
}

```

Dua contoh kode berikut dari penulisan batch menggambarkan verboseness dan kurangnya keamanan tipe saat menggunakan klien standar sebagai lawan dari klien yang ditingkatkan.

**Bandingkan: Batch write menggunakan klien DynamoDB standar**

```

public static void batchWriteStandard(DynamoDbClient dynamoDbClient, String
tableName) {

    Map<String, AttributeValue> catalogItem = Map.of(
        "authors", AttributeValue.builder().ss("a", "b").build(),
        "id", AttributeValue.builder().n("1").build(),
        "isbn", AttributeValue.builder().s("1-565-85698").build(),
        "title", AttributeValue.builder().s("Title 1").build(),
        "price", AttributeValue.builder().n("52.13").build());

    Map<String, AttributeValue> catalogItem2 = Map.of(
        "authors", AttributeValue.builder().ss("a", "b", "c").build(),
        "id", AttributeValue.builder().n("2").build(),
        "isbn", AttributeValue.builder().s("1-208-98073").build(),
        "title", AttributeValue.builder().s("Title 2").build(),
        "price", AttributeValue.builder().n("21.99").build());

    Map<String, AttributeValue> catalogItem3 = Map.of(
        "authors", AttributeValue.builder().ss("g", "k", "c").build(),
        "id", AttributeValue.builder().n("3").build(),
        "isbn", AttributeValue.builder().s("7-236-98618").build(),
        "title", AttributeValue.builder().s("Title 3").build(),
        "price", AttributeValue.builder().n("42.00").build());

    Set<WriteRequest> writeRequests = Set.of(
        WriteRequest.builder().putRequest(b -> b.item(catalogItem)).build(),
        WriteRequest.builder().putRequest(b -> b.item(catalogItem2)).build(),

```

```

        WriteRequest.builder().putRequest(b -> b.item(catalogItem3)).build());

    Map<String, Set<WriteRequest>> productCatalogItems = Map.of(
        "ProductCatalog", writeRequests);

    BatchWriteItemResponse response = dynamoDbClient.batchWriteItem(b ->
    b.requestItems(productCatalogItems));

    logger.info("Unprocessed items: " + response.unprocessedItems().size());
}

```

## Bandingkan: Batch write menggunakan DynamoDB Enhanced Client

```

public static void batchWriteEnhanced(DynamoDbTable<ProductCatalog> productCatalog)
{
    ProductCatalog prod = ProductCatalog.builder()
        .id(1)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(52.13))
        .title("Title 1")
        .build();
    ProductCatalog prod2 = ProductCatalog.builder()
        .id(2)
        .isbn("1-208-98073")
        .authors(new HashSet<>(Arrays.asList("a", "b", "c")))
        .price(BigDecimal.valueOf(21.99))
        .title("Title 2")
        .build();
    ProductCatalog prod3 = ProductCatalog.builder()
        .id(3)
        .isbn("7-236-98618")
        .authors(new HashSet<>(Arrays.asList("g", "k", "c")))
        .price(BigDecimal.valueOf(42.00))
        .title("Title 3")
        .build();

    BatchWriteResult batchWriteResult = DependencyFactory.enhancedClient()
        .batchWriteItem(b -> b.writeBatches(
            WriteBatch.builder(ProductCatalog.class)
                .mappedTableResource(productCatalog)
                .addPutItem(prod).addPutItem(prod2).addPutItem(prod3)
                .build())

```



```
    ));  
    logger.info("Unprocessed items: " +  
batchWriteResult.unprocessedPutItemsForTable(productCatalog).size());  
}
```

Bekerja dengan kelas data yang tidak dapat diubah

Fitur pemetaan DynamoDB Enhanced Client API bekerja dengan kelas data yang tidak dapat diubah. Kelas yang tidak dapat diubah hanya memiliki getter dan memerlukan kelas pembangun yang digunakan SDK untuk membuat instance kelas. Alih-alih menggunakan `@DynamoDbBean` anotasi seperti yang ditunjukkan di [kelas Pelanggan, kelas](#) yang tidak dapat diubah menggunakan `@DynamoDbImmutable` anotasi, yang mengambil parameter yang menunjukkan kelas pembangun untuk digunakan.

Kelas berikut adalah versi yang tidak dapat diubah dari `Customer`

```
package org.example.tests.model.immutable;  
  
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbImmutable;  
import  
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;  
import  
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;  
import  
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;  
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;  
  
import java.time.Instant;  
  
@DynamoDbImmutable(builder = CustomerImmutable.Builder.class)  
public class CustomerImmutable {  
    private final String id;  
    private final String name;  
    private final String email;  
    private final Instant regDate;  
  
    private CustomerImmutable(Builder b) {  
        this.id = b.id;  
        this.email = b.email;  
        this.name = b.name;  
        this.regDate = b.regDate;  
    }  
}
```

```

// This method will be automatically discovered and used by the TableSchema.
public static Builder builder() { return new Builder(); }

@DynamoDbPartitionKey
public String id() { return this.id; }

@DynamoDbSortKey
public String email() { return this.email; }

@DynamoDbSecondaryPartitionKey(indexNames = "customers_by_name")
public String name() { return this.name; }

@DynamoDbSecondarySortKey(indexNames = {"customers_by_date", "customers_by_name"})
public Instant regDate() { return this.regDate; }

public static final class Builder {
    private String id;
    private String email;
    private String name;
    private Instant regDate;

    // The private Builder constructor is visible to the enclosing Customer class.
    private Builder() {}

    public Builder id(String accountId) { this.id = id; return this; }
    public Builder email(String email) { this.email = email; return this; }
    public Builder name(String name) { this.name = name; return this; }
    public Builder regDate(Instant regDate) { this.regDate = regDate; return
this; }

    // This method will be automatically discovered and used by the TableSchema.
    public CustomerImmutable build() { return new CustomerImmutable(this); }
}
}

```

Anda harus memenuhi persyaratan berikut ketika Anda membuat anotasi kelas data dengan `@DynamoDbImmutable`

1. Setiap metode yang keduanya bukan penggantikan `Object.class` dan belum dianotasi `@DynamoDbIgnore` harus menjadi pengambil untuk atribut tabel DynamoDB.
2. Setiap pengambil harus memiliki penyetel case-sensitive yang sesuai pada kelas builder.
3. Hanya satu dari kondisi konstruksi berikut yang harus dipenuhi.

- Kelas builder harus memiliki konstruktor default publik.
  - Kelas data harus memiliki metode statis publik bernama `builder()` yang tidak mengambil parameter dan mengembalikan instance dari kelas builder. Opsi ini ditampilkan di kelas yang tidak dapat diubah `Customer`.
4. Kelas builder harus memiliki metode publik bernama `build()` yang tidak mengambil parameter dan mengembalikan instance dari kelas yang tidak dapat diubah.

Untuk membuat `TableSchema` untuk kelas abadi Anda, gunakan `fromImmutableClass()` metode pada `TableSchema` seperti yang ditunjukkan dalam cuplikan berikut.

```
static final TableSchema<CustomerImmutable> customerImmutableTableSchema =
    TableSchema.fromImmutableClass(CustomerImmutable.class);
```

Sama seperti Anda dapat membuat tabel DynamoDB dari kelas yang bisa berubah, Anda dapat membuatnya dari kelas yang tidak dapat diubah dengan panggilan satu kali `createTable()` ke `DynamoDbTable` dari seperti yang ditunjukkan pada contoh cuplikan berikut.

```
static void createTableFromImmutable(DynamoDbEnhancedClient enhancedClient, String
tableName, DynamoDbWaiter waiter){
    // First, create an in-memory representation of the table using the 'table()'
method of the DynamoDb Enhanced Client.
    // 'table()' accepts a name for the table and a TableSchema instance that you
created previously.
    DynamoDbTable<CustomerImmutable> customerDynamoDbTable = enhancedClient
        .table(tableName, TableSchema.fromImmutableClass(CustomerImmutable.class));

    // Second, call the 'createTable()' method on the DynamoDbTable instance.
    customerDynamoDbTable.createTable();
    waiter.waitUntilTableExists(b -> b.tableName(tableName));
}
```

Gunakan perpustakaan pihak ketiga, seperti Lombok

Perpustakaan pihak ketiga, seperti [Project Lombok](#), membantu menghasilkan kode boilerplate yang terkait dengan objek yang tidak dapat diubah. DynamoDB Enhanced Client API bekerja dengan pustaka ini selama kelas data mengikuti konvensi yang dirinci di bagian ini.

Contoh berikut menunjukkan CustomerImmutable kelas abadi dengan anotasi Lombok. Perhatikan bagaimana onMethod fitur Lombok menyalin anotasi DynamoDB berbasis atribut, seperti, ke kode yang dihasilkan. @DynamoDbPartitionKey

```
@Value
@Builder
@dynamoDbImmutable(builder = Customer.CustomerBuilder.class)
public class Customer {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;

    @Getter(onMethod_=@DynamoDbSortKey)
    private String email;

    @Getter(onMethod_=@DynamoDbSecondaryPartitionKey(indexNames = "customers_by_name"))
    private String name;

    @Getter(onMethod_=@DynamoDbSecondarySortKey(indexNames = {"customers_by_date",
"customers_by_name"}))
    private Instant createdAt;
}
```

Gunakan ekspresi dan kondisi

[Ekspresi dalam DynamoDB Enhanced Client API adalah representasi Java dari ekspresi DynamoDB.](#)

DynamoDB Enhanced Client API menggunakan tiga jenis ekspresi:

### [Ekspresi](#)

ExpressionKelas digunakan ketika Anda menentukan kondisi dan filter.

### [QueryConditional](#)

Jenis ekspresi ini merupakan [kondisi utama](#) untuk operasi kueri.

### [UpdateExpression](#)

Kelas ini membantu Anda menulis ekspresi [pembaruan DynamoDB](#) dan saat ini digunakan dalam kerangka ekstensi saat Anda memperbarui item.

Anatomi ekspresi

Ekspresi terdiri dari yang berikut:

- Ekspresi string (wajib). String berisi ekspresi logika DynamoDB dengan nama placeholder untuk nama atribut dan nilai atribut.
- Peta nilai ekspresi (biasanya diperlukan).
- Peta nama ekspresi (opsional).

Gunakan pembangun untuk menghasilkan Expression objek yang mengambil bentuk umum berikut.

```
Expression expression = Expression.builder()
    .expression(<String>)
    .expressionNames(<Map>)
    .expressionValues(<Map>)
    .build()
```

ExpressionS biasanya membutuhkan peta nilai ekspresi. Peta memberikan nilai untuk placeholder dalam ekspresi string. Kunci peta terdiri dari nama placeholder yang didahului dengan titik dua (: ) dan nilai peta adalah instance dari [AttributeValue](#) [AttributeValues](#) Kelas memiliki metode kenyamanan untuk menghasilkan AttributeValue instance dari literal. Atau, Anda dapat menggunakan AttributeValue.Builder untuk menghasilkan AttributeValue instance.

Cuplikan berikut menunjukkan peta dengan dua entri setelah baris komentar 2. String diteruskan ke expression() metode, ditampilkan setelah baris komentar 1, berisi placeholder yang DynamoDB menyelesaikan sebelum melakukan operasi. Cuplikan ini tidak berisi peta nama ekspresi, karena harga adalah nama atribut yang diizinkan.

```
public static void scanAsync(DynamoDbAsyncTable productCatalog) {
    ScanEnhancedRequest request = ScanEnhancedRequest.builder()
        .consistentRead(true)
        .attributesToProject("id", "title", "authors", "price")
        .filterExpression(Expression.builder()
            // 1. :min_value and :max_value are placeholders for the values
            // provided by the map
            .expression("price >= :min_value AND price <= :max_value")
            // 2. Two values are needed for the expression and each is
            // supplied as a map entry.
            .expressionValues(
                Map.of( ":min_value", numberValue(8.00),
                    ":max_value", numberValue(400_000.00)))
            .build())
    }
```

```
.build();
```

Jika nama atribut dalam tabel DynamoDB adalah kata cadangan, dimulai dengan angka, atau berisi spasi, peta nama ekspresi diperlukan untuk `Expression`

Misalnya, jika nama atribut `1price` bukan `price` dalam contoh kode sebelumnya, contoh akan perlu dimodifikasi seperti yang ditunjukkan pada contoh berikut.

```
ScanEnhancedRequest request = ScanEnhancedRequest.builder()
    .filterExpression(Expression.builder()
        .expression("#price >= :min_value AND #price <= :max_value")
        .expressionNames( Map.of("#price", "1price") )
        .expressionValues(
            Map.of(":min_value", numberValue(8.00),
                ":max_value", numberValue(400_000.00)))
        .build())
    .build();
```

Sebuah placeholder untuk nama ekspresi dimulai dengan tanda pound (`#`). Entri untuk peta nama ekspresi menggunakan placeholder sebagai kunci dan nama atribut sebagai nilai. Peta ditambahkan ke pembuat ekspresi dengan `expressionNames()` metode. DynamoDB menyelesaikan nama atribut sebelum melakukan operasi.

Nilai ekspresi tidak diperlukan jika fungsi digunakan dalam ekspresi string. Contoh fungsi ekspresi adalah `attribute_exists(<attribute_name>)`.

Contoh berikut membangun sebuah `Expression` yang menggunakan fungsi [DynamoDB](#). String ekspresi dalam contoh ini tidak menggunakan placeholder. Ekspresi ini dapat digunakan pada `putItem` operasi untuk memeriksa apakah item sudah ada dalam database dengan nilai `movie` atribut sama dengan `movie` atribut objek data.

```
Expression exp = Expression.builder().expression("attribute_not_exists
(movie)").build();
```

Panduan Pengembang DynamoDB berisi informasi lengkap tentang ekspresi [tingkat rendah yang digunakan dengan DynamoDB](#).

### Ekspresi kondisi dan kondisional

Ketika Anda menggunakan `putItem()`, `updateItem()`, dan `deleteItem()` metode, dan juga ketika Anda menggunakan transaksi dan operasi batch, Anda menggunakan [Expression](#) objek

untuk menentukan kondisi yang DynamoDB harus memenuhi untuk melanjutkan operasi. Ekspresi ini diberi nama ekspresi kondisi. Sebagai contoh, lihat ekspresi kondisi yang digunakan dalam `addDeleteItem()` metode (setelah baris komentar 1) dari [contoh transaksi](#) yang ditunjukkan dalam panduan ini.

Ketika Anda bekerja dengan `query()` metode, suatu kondisi dinyatakan sebagai a [QueryConditional](#). `QueryConditional` kelas memiliki beberapa metode kenyamanan statis yang membantu Anda menulis kriteria yang menentukan item mana yang akan dibaca dari DynamoDB.

Sebagai contoh `QueryConditionals`, lihat contoh kode pertama dari [the section called "Querycontoh metode"](#) bagian panduan ini.

### Ekspresi Filter

Ekspresi filter digunakan dalam operasi pemindaian dan kueri untuk memfilter item yang dikembalikan.

Ekspresi filter diterapkan setelah semua data dibaca dari database, sehingga biaya baca sama seperti jika tidak ada filter. [Panduan Pengembang Amazon DynamoDB memiliki informasi lebih lanjut tentang penggunaan ekspresi filter untuk operasi kueri dan pemindaian.](#)

Contoh berikut menunjukkan ekspresi filter ditambahkan ke permintaan scan. Kriteria membatasi barang yang dikembalikan ke barang dengan harga antara 8.00 dan 80,00 inklusif.

```
Map<String, AttributeValue> expressionValues = Map.of(
    ":min_value", numberValue(8.00),
    ":max_value", numberValue(80.00));

ScanEnhancedRequest request = ScanEnhancedRequest.builder()
    .consistentRead(true)
    // 1. the 'attributesToProject()' method allows you to specify which
values you want returned.
    .attributesToProject("id", "title", "authors", "price")
    // 2. Filter expression limits the items returned that match the
provided criteria.
    .filterExpression(Expression.builder()
        .expression("price >= :min_value AND price <= :max_value")
        .expressionValues(expressionValues)
        .build())
    .build();
```

## Perbarui ekspresi

Metode DynamoDB Enhanced Client `updateItem()` menyediakan cara standar untuk memperbarui item di DynamoDB. [Namun, ketika Anda memerlukan lebih banyak fungsionalitas, UpdateExpressions](#) berikan representasi `type-safe` dari sintaks ekspresi pembaruan DynamoDB. Misalnya, Anda dapat menggunakan `UpdateExpressions` untuk meningkatkan nilai tanpa terlebih dahulu membaca item dari DynamoDB, atau menambahkan anggota individu ke daftar. Ekspresi pembaruan saat ini tersedia di ekstensi khusus untuk `updateItem()` metode ini.

Untuk contoh yang menggunakan ekspresi pembaruan, lihat [contoh ekstensi kustom](#) dalam panduan ini.

Informasi selengkapnya tentang ekspresi pembaruan tersedia di Panduan Pengembang [Amazon DynamoDB](#).

## Bekerja dengan hasil paginasi: pemindaian dan kueri

Metode `scan`, `query` dan `batch` metode DynamoDB Enhanced Client API mengembalikan respons dengan satu halaman atau beberapa. Sebuah halaman berisi satu atau lebih item. Kode Anda dapat memproses respons per halaman atau dapat memproses item individual.

Respons paginasi yang dikembalikan oleh `DynamoDbEnhancedClient` klien sinkron mengembalikan [PageIterable](#) objek, sedangkan respons yang dikembalikan oleh `DynamoDbEnhancedAsyncClient` asinkron mengembalikan objek [PagePublisher](#)

Bagian ini membahas pemrosesan hasil paginasi dan memberikan contoh yang menggunakan API pemindaian dan kueri.

## Memindai tabel

[scan](#) Metode SDK sesuai dengan operasi [DynamoDB](#) dengan nama yang sama. DynamoDB Enhanced Client API menawarkan opsi yang sama tetapi menggunakan model objek yang sudah dikenal dan menangani pagination untuk Anda.

Pertama, kita menjelajahi `PageIterable` antarmuka dengan melihat `scan` metode kelas pemetaan sinkron, [DynamoDbTable](#)

## Gunakan API sinkron

Contoh berikut menunjukkan `scan` metode yang menggunakan [ekspresi](#) untuk memfilter item yang dikembalikan. [ProductCatalog](#) ini adalah objek model yang ditunjukkan sebelumnya.



Ekspresi pemfilteran yang ditampilkan setelah baris komentar 1 membatasi ProductCatalog item yang dikembalikan ke item dengan nilai harga antara 8,00 dan 80,00 secara inklusif.

Contoh ini juga mengecualikan isbn nilai dengan menggunakan attributesToProject metode yang ditampilkan setelah baris komentar 2.

Pada baris komentar 3, PageIterable objekpagedResult, dikembalikan oleh scan metode. streamMetode PageIterable mengembalikan [java.util.Stream](#) objek, yang dapat Anda gunakan untuk memproses halaman. Dalam contoh ini, jumlah halaman dihitung dan dicatat.

Dimulai dengan baris komentar 4, contoh menunjukkan dua variasi mengakses ProductCatalog item. Versi setelah baris komentar 2a mengalir melalui setiap halaman dan mengurutkan dan mencatat item di setiap halaman. Versi setelah baris komentar 2b melewati iterasi halaman dan mengakses item secara langsung.

PageIterableAntarmuka menawarkan beberapa cara untuk memproses hasil karena dua antarmuka induknya— [java.lang.Iterable](#) dan [SdkIterable](#). IterablemembawaforEach, iterator dan spliterator metode, dan SdkIterable membawa stream metode.

```
public static void scanSync(DynamoDbTable<ProductCatalog> productCatalog) {

    Map<String, AttributeValue> expressionValues = Map.of(
        ":min_value", numberValue(8.00),
        ":max_value", numberValue(80.00));

    ScanEnhancedRequest request = ScanEnhancedRequest.builder()
        .consistentRead(true)
        // 1. the 'attributesToProject()' method allows you to specify which
values you want returned.
        .attributesToProject("id", "title", "authors", "price")
        // 2. Filter expression limits the items returned that match the
provided criteria.
        .filterExpression(Expression.builder()
            .expression("price >= :min_value AND price <= :max_value")
            .expressionValues(expressionValues)
            .build())
        .build();

    // 3. A PageIterable object is returned by the scan method.
    PageIterable<ProductCatalog> pagedResults = productCatalog.scan(request);
    logger.info("page count: {}", pagedResults.stream().count());
}
```

```

// 4. Log the returned ProductCatalog items using two variations.
// 4a. This version sorts and logs the items of each page.
pagedResults.stream().forEach(p -> p.items().stream()
    .sorted(Comparator.comparing(ProductCatalog::price))
    .forEach(
        item -> logger.info(item.toString())
    ));
// 4b. This version sorts and logs all items for all pages.
pagedResults.items().stream()
    .sorted(Comparator.comparing(ProductCatalog::price))
    .forEach(
        item -> logger.info(item.toString())
    );
}

```

## Gunakan API asinkron

scanMetode asinkron mengembalikan hasil sebagai objek. PagePublisher PagePublisherAntarmuka memiliki dua subscribe metode yang dapat Anda gunakan untuk memproses halaman respons. Salah satu subscribe metode berasal dari antarmuka org.reactivestreams.Publisher induk. Untuk memproses halaman menggunakan opsi pertama ini, berikan [Subscriber](#) instance ke subscribe metode. Contoh pertama yang berikut menunjukkan penggunaan subscribe metode.

subscribeMetode kedua berasal dari [SdkPublisher](#) antarmuka. Versi ini subscribe menerima [Consumer](#) bukan aSubscriber. Variasi subscribe metode ini ditunjukkan pada contoh kedua berikut.

Contoh berikut menunjukkan versi asinkron dari scan metode yang menggunakan ekspresi filter yang sama yang ditunjukkan pada contoh sebelumnya.

Setelah baris komentar 3, DynamoDbAsyncTable.scan mengembalikan PagePublisher objek. Pada baris berikutnya, kode membuat instance org.reactivestreams.Subscriber antarmuka, ProductCatalogSubscriber, yang berlangganan baris PagePublisher setelah komentar 4.

SubscriberObjek mengumpulkan ProductCatalog item dari setiap halaman dalam onNext metode setelah baris komentar 8 dalam contoh ProductCatalogSubscriber kelas. Item disimpan dalam List variabel pribadi dan diakses dalam kode panggilan dengan ProductCatalogSubscriber.getSubscribedItems() metode. Ini disebut setelah baris komentar 5.

Setelah daftar diambil, kode mengurutkan semua ProductCatalog item berdasarkan harga dan mencatat setiap item.

[CountDownLatch](#) Di ProductCatalogSubscriber kelas memblokir thread panggilan sampai semua item telah ditambahkan ke daftar sebelum melanjutkan setelah baris komentar 5.

```
public static void scanAsync(DynamoDbAsyncTable productCatalog) {
    ScanEnhancedRequest request = ScanEnhancedRequest.builder()
        .consistentRead(true)
        .attributesToProject("id", "title", "authors", "price")
        .filterExpression(Expression.builder()
            // 1. :min_value and :max_value are placeholders for the values
            // provided by the map
            .expression("price >= :min_value AND price <= :max_value")
            // 2. Two values are needed for the expression and each is
            // supplied as a map entry.
            .expressionValues(
                Map.of( ":min_value", numberValue(8.00),
                    ":max_value", numberValue(400_000.00)))
            .build())
        .build();

    // 3. A PagePublisher object is returned by the scan method.
    PagePublisher<ProductCatalog> pagePublisher = productCatalog.scan(request);
    ProductCatalogSubscriber subscriber = new ProductCatalogSubscriber();
    // 4. Subscribe the ProductCatalogSubscriber to the PagePublisher.
    pagePublisher.subscribe(subscriber);
    // 5. Retrieve all collected ProductCatalog items accumulated by the
    subscriber.
    subscriber.getSubscribedItems().stream()
        .sorted(Comparator.comparing(ProductCatalog::price))
        .forEach(item ->
            logger.info(item.toString()));
    // 6. Use a Consumer to work through each page.
    pagePublisher.subscribe(page -> page
        .items().stream()
        .sorted(Comparator.comparing(ProductCatalog::price))
        .forEach(item ->
            logger.info(item.toString())))
        .join(); // If needed, blocks the subscribe() method thread until it is
    finished processing.
    // 7. Use a Consumer to work through each ProductCatalog item.
    pagePublisher.items()
}
```

```

        .subscribe(product -> logger.info(product.toString()))
        .exceptionally(failure -> {
            logger.error("ERROR - ", failure);
            return null;
        })
        .join(); // If needed, blocks the subscribe() method thread until it is
finished processing.
    }

```

```

private static class ProductCatalogSubscriber implements
Subscriber<Page<ProductCatalog>> {
    private CountDownLatch latch = new CountDownLatch(1);
    private Subscription subscription;
    private List<ProductCatalog> itemsFromAllPages = new ArrayList<>();

    @Override
    public void onSubscribe(Subscription sub) {
        subscription = sub;
        subscription.request(1L);
        try {
            latch.await(); // Called by main thread blocking it until latch is
released.
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    public void onNext(Page<ProductCatalog> productCatalogPage) {
        // 8. Collect all the ProductCatalog instances in the page, then ask the
publisher for one more page.
        itemsFromAllPages.addAll(productCatalogPage.items());
        subscription.request(1L);
    }

    @Override
    public void onError(Throwable throwable) {
    }

    @Override
    public void onComplete() {
        latch.countDown(); // Call by subscription thread; latch releases.
    }
}

```

```

    List<ProductCatalog> getSubscribedItems() {
        return this.itemsFromAllPages;
    }
}

```

Contoh cuplikan berikut menggunakan versi `PagePublisher.subscribe` metode yang menerima baris `Consumer` setelah komentar 6. Parameter lambda Java mengkonsumsi halaman, yang selanjutnya memproses setiap item. Dalam contoh ini, setiap halaman diproses dan item pada setiap halaman diurutkan dan kemudian dicatat.

```

// 6. Use a Consumer to work through each page.
pagePublisher.subscribe(page -> page
    .items().stream()
    .sorted(Comparator.comparing(ProductCatalog::price))
    .forEach(item ->
        logger.info(item.toString())))
    .join(); // If needed, blocks the subscribe() method thread until it is
finished processing.

```

`items` Metode `PagePublisher` membuka bungkus contoh model sehingga kode Anda dapat memproses item secara langsung. Pendekatan ini ditunjukkan dalam cuplikan berikut.

```

// 7. Use a Consumer to work through each ProductCatalog item.
pagePublisher.items()
    .subscribe(product -> logger.info(product.toString()))
    .exceptionally(failure -> {
        logger.error("ERROR - ", failure);
        return null;
    })
    .join(); // If needed, blocks the subscribe() method thread until it is
finished processing.

```

## Mengkueri Tabel

[`query\(\)`](#) Metode `DynamoDbTable` kelas menemukan item berdasarkan nilai kunci primer. `@DynamoDbPartitionKey` anotasi dan `@DynamoDbSortKey` anotasi opsional digunakan untuk menentukan kunci utama pada kelas data Anda.

`query()` Metode ini membutuhkan nilai kunci partisi yang menemukan item yang cocok dengan nilai yang diberikan. Jika tabel Anda juga mendefinisikan kunci pengurutan, Anda dapat menambahkan

nilai untuk itu ke kueri Anda sebagai kondisi perbandingan tambahan untuk menyempurnakan hasilnya.

Kecuali untuk memproses hasil, versi sinkron dan asinkron bekerja sama. `query()` Seperti halnya scan API, query API mengembalikan panggilan `PageIterable` untuk sinkron dan panggilan asinkron `PagePublisher` untuk. Kami membahas penggunaan `PageIterable` dan `PagePublisher` sebelumnya di bagian pemindaian.

## Query contoh metode

Contoh kode `query()` metode yang mengikuti menggunakan `MovieActor` kelas. Kelas data mendefinisikan kunci primer komposit yang terdiri dari `movie` atribut untuk kunci partisi dan `actor` atribut untuk kunci sortir.

Kelas ini juga memberi sinyal bahwa ia menggunakan indeks sekunder global bernama `acting_award_year`. Kunci primer komposit indeks terdiri dari `actingaward` atribut untuk kunci partisi dan `actingyear` untuk kunci sortir. Kemudian dalam topik ini, ketika kita menunjukkan cara membuat dan menggunakan indeks, kita akan merujuk ke `acting_award_year` indeks.

## MovieActor kelas

```
package org.example.tests.model;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbAttribute;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.util.Objects;

@DynamoDbBean
public class MovieActor implements Comparable<MovieActor> {

    private String movieName;
    private String actorName;
    private String actingAward;
    private Integer actingYear;
```

```
private String actingSchoolName;

@DynamoDbPartitionKey
@DynamoDbAttribute("movie")
public String getMovieName() {
    return movieName;
}

public void setMovieName(String movieName) {
    this.movieName = movieName;
}

@DynamoDbSortKey
@DynamoDbAttribute("actor")
public String getActorName() {
    return actorName;
}

public void setActorName(String actorName) {
    this.actorName = actorName;
}

@DynamoDbSecondaryPartitionKey(indexNames = "acting_award_year")
@DynamoDbAttribute("actingaward")
public String getActingAward() {
    return actingAward;
}

public void setActingAward(String actingAward) {
    this.actingAward = actingAward;
}

@DynamoDbSecondarySortKey(indexNames = {"acting_award_year", "movie_year"})
@DynamoDbAttribute("actingyear")
public Integer getActingYear() {
    return actingYear;
}

public void setActingYear(Integer actingYear) {
    this.actingYear = actingYear;
}

@DynamoDbAttribute("actingschoolname")
public String getActingSchoolName() {
```

```
        return actingSchoolName;
    }

    public void setActingSchoolName(String actingSchoolName) {
        this.actingSchoolName = actingSchoolName;
    }

    @Override
    public String toString() {
        final StringBuffer sb = new StringBuffer("MovieActor{");
        sb.append("movieName=").append(movieName).append('\ ');
        sb.append(", actorName=").append(actorName).append('\ ');
        sb.append(", actingAward=").append(actingAward).append('\ ');
        sb.append(", actingYear=").append(actingYear);
        sb.append(", actingSchoolName=").append(actingSchoolName).append('\ ');
        sb.append('}');
        return sb.toString();
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        MovieActor that = (MovieActor) o;
        return Objects.equals(movieName, that.movieName) && Objects.equals(actorName,
that.actorName) && Objects.equals(actingAward, that.actingAward) &&
Objects.equals(actingYear, that.actingYear) && Objects.equals(actingSchoolName,
that.actingSchoolName);
    }

    @Override
    public int hashCode() {
        return Objects.hash(movieName, actorName, actingAward, actingYear,
actingSchoolName);
    }

    @Override
    public int compareTo(MovieActor o) {
        if (this.movieName.compareTo(o.movieName) != 0){
            return this.movieName.compareTo(o.movieName);
        } else {
            return this.actorName.compareTo(o.actorName);
        }
    }
}
```



```
}
```

Contoh kode yang mengikuti kueri terhadap item berikut.

### Item dalam **MovieActor** tabel

```
MovieActor{movieName='movie01', actorName='actor0', actingAward='actingaward0',
  actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
  actingYear=2001, actingSchoolName='actingschool1'}
MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
  actingYear=2001, actingSchoolName='actingschool2'}
MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',
  actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
  actingYear=2001, actingSchoolName='actingschool4'}
MovieActor{movieName='movie02', actorName='actor0', actingAward='actingaward0',
  actingYear=2002, actingSchoolName='null'}
MovieActor{movieName='movie02', actorName='actor1', actingAward='actingaward1',
  actingYear=2002, actingSchoolName='actingschool1'}
MovieActor{movieName='movie02', actorName='actor2', actingAward='actingaward2',
  actingYear=2002, actingSchoolName='actingschool2'}
MovieActor{movieName='movie02', actorName='actor3', actingAward='actingaward3',
  actingYear=2002, actingSchoolName='null'}
MovieActor{movieName='movie02', actorName='actor4', actingAward='actingaward4',
  actingYear=2002, actingSchoolName='actingschool4'}
MovieActor{movieName='movie03', actorName='actor0', actingAward='actingaward0',
  actingYear=2003, actingSchoolName='null'}
MovieActor{movieName='movie03', actorName='actor1', actingAward='actingaward1',
  actingYear=2003, actingSchoolName='actingschool1'}
MovieActor{movieName='movie03', actorName='actor2', actingAward='actingaward2',
  actingYear=2003, actingSchoolName='actingschool2'}
MovieActor{movieName='movie03', actorName='actor3', actingAward='actingaward3',
  actingYear=2003, actingSchoolName='null'}
MovieActor{movieName='movie03', actorName='actor4', actingAward='actingaward4',
  actingYear=2003, actingSchoolName='actingschool4'}
```

Kode berikut mendefinisikan dua [QueryConditional](#) contoh. `QueryConditionals` bekerja dengan nilai kunci—baik kunci partisi saja atau dalam kombinasi dengan kunci pengurutan—dan sesuai dengan [ekspresi kondisional kunci](#) dari API layanan DynamoDB. Setelah baris komentar 1, contoh mendefinisikan `keyEqual` instance yang cocok dengan item dengan nilai partisi. **movie01**

Contoh ini juga mendefinisikan ekspresi filter yang memfilter item apa pun yang tidak **actingschoolname** aktif setelah baris komentar 2.

Setelah baris komentar 3, contoh menunjukkan [QueryEnhancedRequest](#) contoh bahwa kode diteruskan ke `DynamoDbTable.query()` metode. Objek ini menggabungkan kondisi kunci dan filter yang digunakan SDK untuk menghasilkan permintaan ke layanan DynamoDB.

```
public static void query(DynamoDbTable movieActorTable) {

    // 1. Define a QueryConditional instance to return items matching a partition
    value.
    QueryConditional keyEqual = QueryConditional.keyEqualTo(b ->
b.partitionValue("movie01"));
    // 1a. Define a QueryConditional that adds a sort key criteria to the partition
    value criteria.
    QueryConditional sortGreaterThanOrEqualTo =
QueryConditional.sortGreaterThanOrEqualTo(b ->
b.partitionValue("movie01").sortValue("actor2"));
    // 2. Define a filter expression that filters out items whose attribute value
    is null.
    final Expression filterOutNoActingschoolname =
Expression.builder().expression("attribute_exists(actingschoolname)").build();

    // 3. Build the query request.
    QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
        .queryConditional(keyEqual)
        .filterExpression(filterOutNoActingschoolname)
        .build();
    // 4. Perform the query.
    PageIterable<MovieActor> pagedResults = movieActorTable.query(tableQuery);
    logger.info("page count: {}", pagedResults.stream().count()); // Log number of
    pages.

    pagedResults.items().stream()
        .sorted()
        .forEach(
            item -> logger.info(item.toString()) // Log the sorted list of
            items.
        );
}
```

Berikut ini adalah output dari menjalankan metode. Output menampilkan item dengan `movieName` nilai `movie01` dan tidak menampilkan item yang `actingSchoolName` sama dengan **null**

```

2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:46 - page count: 1
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
  actingYear=2001, actingSchoolName='actingschool1'}
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
  actingYear=2001, actingSchoolName='actingschool2'}
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
  actingYear=2001, actingSchoolName='actingschool4'}

```

Dalam variasi permintaan kueri berikut yang ditampilkan sebelumnya setelah baris komentar 3, kode menggantikan `keyEqual QueryConditional` dengan `sortGreaterThanOrEqualTo QueryConditional` yang didefinisikan setelah baris komentar 1a. Kode berikut juga menghapus ekspresi filter.

```

QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
    .queryConditional(sortGreaterThanOrEqualTo)

```

Karena tabel ini memiliki kunci primer komposit, semua `QueryConditional` instance memerlukan nilai kunci partisi. `QueryConditional` metode yang dimulai dengan `sort...` menunjukkan bahwa kunci pengurutan diperlukan. Hasilnya tidak diurutkan.

Output berikut menampilkan hasil dari query. Kueri mengembalikan item yang memiliki **movieName** nilai sama dengan `movie01` dan hanya item yang memiliki **actorName** nilai yang lebih besar dari atau sama dengan `actor2`. Karena filter telah dihapus, kueri mengembalikan item yang tidak memiliki nilai untuk `actingSchoolName` atribut.

```

2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:46 - page count: 1
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
  actingYear=2001, actingSchoolName='actingschool2'}
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',
  actingYear=2001, actingSchoolName='null'}
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:51 -
  MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
  actingYear=2001, actingSchoolName='actingschool4'}

```

## Lakukan operasi batch

[DynamoDB Enhanced Client API menawarkan dua metode batch `batchGetItem\(\)` dan `batchWriteItem\(\)`](#).

### Contoh `batchGetItem()`

Dengan [`DynamoDbTable.batchGetItem\(\)`](#) metode ini, Anda dapat mengambil hingga 100 item individual di beberapa tabel dalam satu permintaan keseluruhan. Contoh berikut menggunakan kelas [Customer](#) dan [MovieActor](#) data yang ditunjukkan sebelumnya.

Dalam contoh setelah baris 1 dan 2, Anda membangun [ReadBatch](#) objek yang kemudian Anda tambahkan sebagai parameter ke `batchGetItem()` metode setelah baris komentar 3. Kode setelah baris komentar 1 membangun batch untuk dibaca dari Customer tabel. Kode setelah baris komentar 1a menunjukkan penggunaan [GetItemEnhancedRequest](#) pembangun yang mengambil nilai kunci primer untuk menentukan item yang akan dibaca. Berbeda dengan menentukan nilai kunci untuk meminta item, Anda dapat menggunakan kelas data untuk meminta item seperti yang ditunjukkan setelah baris komentar 1b. SDK mengekstrak nilai kunci di balik layar sebelum mengirimkan permintaan.

[Saat Anda menentukan item menggunakan pendekatan berbasis kunci seperti yang ditunjukkan dalam dua pernyataan setelah 2a, Anda juga dapat menentukan bahwa DynamoDB harus melakukan pembacaan yang sangat konsisten.](#) Ketika `consistentRead()` metode ini digunakan, itu harus digunakan pada semua item yang diminta untuk tabel yang sama.

Untuk mengambil item yang DynamoDB temukan, gunakan metode [resultsForTable\(\)](#) yang ditampilkan setelah baris komentar 4. Panggil metode untuk setiap tabel yang dibaca dalam permintaan. `resultsForTable()` mengembalikan daftar item yang ditemukan yang dapat Anda proses menggunakan `java.util.List` metode apapun. Contoh ini mencatat setiap item.

Untuk menemukan item yang DynamoDB tidak memproses, gunakan pendekatan setelah baris komentar 5. `BatchGetResultPageKelas` memiliki [unprocessedKeysForTable\(\)](#) metode yang memberi Anda akses ke setiap kunci yang belum diproses. [Referensi BatchGetItem API](#) memiliki informasi lebih lanjut tentang situasi yang menghasilkan item yang tidak diproses.

```
public static void batchGetItemExample(DynamoDbEnhancedClient enhancedClient,
                                      DynamoDbTable<Customer> customerTable,
                                      DynamoDbTable<MovieActor> movieActorTable) {

    Customer customer2 = new Customer();
    customer2.setId("2");
```

```
customer2.setEmail("cust2@example.org");

// 1. Build a batch to read from the Customer table.
ReadBatch customerBatch = ReadBatch.builder(Customer.class)
    .mappedTableResource(customerTable)
    // 1a. Specify the primary key values for the item.
    .addGetItem(b -> b.key(k ->
k.partitionValue("1").sortValue("cust1@orgname.org")))
    // 1b. Alternatively, supply a data class instances to provide the
primary key values.
    .addGetItem(customer2)
    .build();

// 2. Build a batch to read from the MovieActor table.
ReadBatch moveActorBatch = ReadBatch.builder(MovieActor.class)
    .mappedTableResource(movieActorTable)
    // 2a. Call consistentRead(Boolean.TRUE) for each item for the same
table.
    .addGetItem(b -> b.key(k ->
k.partitionValue("movie01").sortValue("actor1")))
    .consistentRead(Boolean.TRUE))
    .addGetItem(b -> b.key(k ->
k.partitionValue("movie01").sortValue("actor4")))
    .consistentRead(Boolean.TRUE))
    .build();

// 3. Add ReadBatch objects to the request.
BatchGetResultPageIterable resultPages = enhancedClient.batchGetItem(b ->
b.readBatches(customerBatch, moveActorBatch));

// 4. Retrieve the successfully requested items from each table.
resultPages.resultsForTable(customerTable).forEach(item ->
logger.info(item.toString()));
resultPages.resultsForTable(movieActorTable).forEach(item ->
logger.info(item.toString()));

// 5. Retrieve the keys of the items requested but not processed by the
service.
resultPages.forEach((BatchGetResultPage pageResult) -> {
    pageResult.unprocessedKeysForTable(customerTable).forEach(key ->
logger.info("Unprocessed item key: " + key.toString()));
    pageResult.unprocessedKeysForTable(movieActorTable).forEach(key ->
logger.info("Unprocessed item key: " + key.toString()));
});
}
```

Asumsikan bahwa item berikut ada di dua tabel sebelum menjalankan kode contoh.

Item dalam tabel

```
Customer [id=1, name=CustName1, email=cust1@example.org,
  regDate=2023-03-31T15:46:27.688Z]
Customer [id=2, name=CustName2, email=cust2@example.org,
  regDate=2023-03-31T15:46:28.688Z]
Customer [id=3, name=CustName3, email=cust3@example.org,
  regDate=2023-03-31T15:46:29.688Z]
Customer [id=4, name=CustName4, email=cust4@example.org,
  regDate=2023-03-31T15:46:30.688Z]
Customer [id=5, name=CustName5, email=cust5@example.org,
  regDate=2023-03-31T15:46:31.689Z]
MovieActor{movieName='movie01', actorName='actor0', actingAward='actingaward0',
  actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
  actingYear=2001, actingSchoolName='actingschool1'}
MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
  actingYear=2001, actingSchoolName='actingschool2'}
MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',
  actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
  actingYear=2001, actingSchoolName='actingschool4'}
```

Output berikut menunjukkan item yang dikembalikan dan dicatat setelah baris komentar 4.

```
Customer [id=1, name=CustName1, email=cust1@example.org,
  regDate=2023-03-31T15:46:27.688Z]
Customer [id=2, name=CustName2, email=cust2@example.org,
  regDate=2023-03-31T15:46:28.688Z]
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
  actingYear=2001, actingSchoolName='actingschool4'}
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
  actingYear=2001, actingSchoolName='actingschool1'}
```

### Contoh `batchWriteItem()`

`batchWriteItem()` Metode ini menempatkan atau menghapus beberapa item dalam satu atau lebih tabel. Anda dapat menentukan hingga 25 operasi put atau hapus individu dalam permintaan. Contoh berikut menggunakan kelas [ProductCatalog](#) dan [MovieActor](#) model yang ditunjukkan sebelumnya.

`WriteBatch` objek dibangun setelah baris komentar 1 dan 2. Untuk `ProductCatalog` tabel, kode menempatkan satu item dan menghapus satu item. Untuk `MovieActor` tabel setelah baris komentar 2, kode menempatkan dua item dan menghapus satu.

`batchWriteItem` metode ini disebut setelah baris komentar 3. [builder](#) parameter menyediakan permintaan batch untuk setiap tabel.

[BatchWriteResult](#) objek yang dikembalikan menyediakan metode terpisah untuk setiap operasi untuk melihat permintaan yang belum diproses. Kode setelah baris komentar 4a menyediakan kunci untuk permintaan penghapusan yang belum diproses dan kode setelah baris komentar 4b menyediakan item put yang belum diproses.

```
public static void batchWriteItemExample(DynamoDbEnhancedClient enhancedClient,
                                         DynamoDbTable<ProductCatalog>
catalogTable,
                                         DynamoDbTable<MovieActor> movieActorTable)
{
    // 1. Build a batch to write to the ProductCatalog table.
    WriteBatch products = WriteBatch.builder(ProductCatalog.class)
        .mappedTableResource(catalogTable)
        .addPutItem(b -> b.item(getProductCatItem1()))
        .addDeleteItem(b -> b.key(k -> k
            .partitionValue(getProductCatItem2().id())
            .sortValue(getProductCatItem2().title())))
        .build();

    // 2. Build a batch to write to the MovieActor table.
    WriteBatch movies = WriteBatch.builder(MovieActor.class)
        .mappedTableResource(movieActorTable)
        .addPutItem(getMovieActorYeoh())
        .addPutItem(getMovieActorBlanchettPartial())
        .addDeleteItem(b -> b.key(k -> k
            .partitionValue(getMovieActorStreep().getMovieName())
            .sortValue(getMovieActorStreep().getActorName())))
        .build();

    // 3. Add WriteBatch objects to the request.
    BatchWriteResult batchWriteResult = enhancedClient.batchWriteItem(b ->
b.writeBatches(products, movies));
    // 4. Retrieve keys for items the service did not process.
    // 4a. 'unprocessedDeleteItemsForTable()' returns keys for delete requests that
did not process.
```

```

        if (batchWriteResult.unprocessedDeleteItemsForTable(movieActorTable).size() >
0) {

batchWriteResult.unprocessedDeleteItemsForTable(movieActorTable).forEach(key ->
        logger.info(key.toString()));
    }
    // 4b. 'unprocessedPutItemsForTable()' returns keys for put requests that did
not process.
    if (batchWriteResult.unprocessedPutItemsForTable(catalogTable).size() > 0) {
        batchWriteResult.unprocessedPutItemsForTable(catalogTable).forEach(key ->
            logger.info(key.toString()));
    }
}

```

Metode pembantu berikut menyediakan objek model untuk operasi put dan delete.

### Metode pembantu

```

public static ProductCatalog getProductCatItem1() {
    return ProductCatalog.builder()
        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
}

public static ProductCatalog getProductCatItem2() {
    return ProductCatalog.builder()
        .id(4)
        .price(BigDecimal.valueOf(40.00))
        .title("Title 1")
        .build();
}

public static MovieActor getMovieActorBlanchettPartial() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Cate Blanchett");
    movieActor.setMovieName("Blue Jasmine");
    movieActor.setActingYear(2023);
    movieActor.setActingAward("Best Actress");
    return movieActor;
}

```



```

public static MovieActor getMovieActorStreep() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Meryl Streep");
    movieActor.setMovieName("Sophie's Choice");
    movieActor.setActingYear(1982);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("Yale School of Drama");
    return movieActor;
}

public static MovieActor getMovieActorYeoh(){
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Michelle Yeoh");
    movieActor.setMovieName("Everything Everywhere All at Once");
    movieActor.setActingYear(2023);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("Royal Academy of Dance");
    return movieActor;
}

```

Asumsikan bahwa tabel berisi item berikut sebelum Anda menjalankan kode contoh.

```

MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}

```

Setelah kode contoh selesai, tabel berisi item berikut.

```

MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='null'}
MovieActor{movieName='Everything Everywhere All at Once', actorName='Michelle Yeoh', actingAward='Best Actress', actingYear=2023, actingSchoolName='Royal Academy of Dance'}
ProductCatalog{id=2, title='Title 55', isbn='1-565-85698', authors=[a, b], price=30.22}

```

Perhatikan dalam `MovieActor` tabel bahwa item `Blue Jasmine` film telah diganti dengan item yang digunakan dalam permintaan `put` yang diperoleh melalui metode `getMovieActorBlanchettPartial()` pembantu. Jika nilai atribut kacang data tidak disediakan,

nilai dalam database dihapus. Inilah sebabnya mengapa hasilnya `actingSchoolName` nol untuk item `Blue Jasmine` film.

### Note

Meskipun dokumentasi API menunjukkan bahwa ekspresi kondisi dapat digunakan dan bahwa kapasitas yang dikonsumsi serta metrik pengumpulan dapat dikembalikan dengan permintaan `put` dan `delete` individual, ini tidak terjadi dalam skenario penulisan batch. Untuk meningkatkan kinerja operasi batch, opsi individual ini diabaikan.

Lakukan operasi transaksi

DynamoDB Enhanced Client API menyediakan dan metode `transactGetItems()`. `transactWriteItems()` Metode transaksi SDK for Java memberikan atomisitas, konsistensi, isolasi, dan daya tahan (ACID) dalam tabel DynamoDB, membantu Anda menjaga kebenaran data dalam aplikasi Anda.

### Contoh `transactGetItems()`

`transactGetItems()` Metode ini menerima hingga 100 permintaan individu untuk item. Semua item dibaca dalam satu transaksi atom. Panduan Pengembang Amazon DynamoDB memiliki informasi tentang [kondisi yang menyebabkan transactGetItems\(\) metode gagal](#), dan juga tentang tingkat isolasi yang digunakan saat Anda menelepon. [transactGetItem\(\)](#)

Setelah baris komentar 1 dalam contoh berikut, kode memanggil `transactGetItems()` metode dengan `builder` parameter. Builder `addGetItem()` dipanggil tiga kali dengan objek data yang berisi nilai kunci yang akan digunakan SDK untuk menghasilkan permintaan akhir.

Permintaan mengembalikan daftar `Document` objek setelah baris komentar 2. Daftar dokumen yang dikembalikan berisi contoh `dokumen` non-null dari data item dalam urutan yang sama seperti yang diminta. `Document.getItem(MappedTableResource<T> mappedTableResource)` Metode mengkonversi objek untyped ke `Document` objek Java diketik jika data item dikembalikan, jika tidak metode mengembalikan null.

```
public static void transactGetItemsExample(DynamoDbEnhancedClient enhancedClient,
                                          DynamoDbTable<ProductCatalog>
catalogTable,
                                          DynamoDbTable<MovieActor>
movieActorTable) {
```

```

// 1. Request three items from two tables using a builder.
final List<Document> documents = enhancedClient.transactGetItems(b -> b
    .addGetItem(catalogTable,
Key.builder().partitionValue(2).sortValue("Title 55").build())
    .addGetItem(movieActorTable, Key.builder().partitionValue("Sophie's
Choice").sortValue("Meryl Streep").build())
    .addGetItem(movieActorTable, Key.builder().partitionValue("Blue
Jasmine").sortValue("Cate Blanchett").build())
    .build());

// 2. A list of Document objects is returned in the same order as requested.
ProductCatalog title55 = documents.get(0).getItem(catalogTable);
if (title55 != null) {
    logger.info(title55.toString());
}

MovieActor sophiesChoice = documents.get(1).getItem(movieActorTable);
if (sophiesChoice != null) {
    logger.info(sophiesChoice.toString());
}

// 3. The getItem() method returns null if the Document object contains no item
from DynamoDB.
MovieActor blueJasmine = documents.get(2).getItem(movieActorTable);
if (blueJasmine != null) {
    logger.info(blueJasmine.toString());
}
}

```

Tabel DynamoDB berisi item berikut sebelum contoh kode berjalan.

```

ProductCatalog{id=2, title='Title 55', isbn='orig_isbn', authors=[b, g], price=10}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}

```

Output berikut dicatat. Jika item diminta tetapi tidak ditemukan, itu tidak dikembalikan seperti halnya permintaan untuk film bernama Blue Jasmine.

```

ProductCatalog{id=2, title='Title 55', isbn='orig_isbn', authors=[b, g], price=10}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}

```

## `transactWriteItems()` contoh

[transactWriteItems\(\)](#) Menerima hingga 100 tindakan put, update, atau delete dalam satu transaksi atom di beberapa tabel. Panduan Pengembang Amazon DynamoDB berisi rincian tentang pembatasan dan kondisi kegagalan operasi layanan [DynamoDB yang mendasarinya](#).

### Contoh basic

Dalam contoh berikut, empat operasi diminta untuk dua tabel. Kelas model yang sesuai [ProductCatalog](#) dan ditunjukkan [MovieActor](#) sebelumnya.

Masing-masing dari tiga kemungkinan operasi—put, update, dan delete—menggunakan parameter permintaan khusus untuk menentukan detailnya.

Kode setelah baris komentar 1 menunjukkan variasi sederhana dari `addPutItem()` metode ini. Metode ini menerima [MappedTableResource](#) objek dan contoh objek data untuk dimasukkan. Pernyataan setelah komentar baris 2 menunjukkan variasi yang menerima [TransactPutItemEnhancedRequest](#) instance. Variasi ini memungkinkan Anda menambahkan lebih banyak opsi dalam permintaan, seperti ekspresi kondisi. [Contoh](#) selanjutnya menunjukkan ekspresi kondisi untuk operasi individu.

Operasi pembaruan diminta setelah baris komentar 3.

[TransactUpdateItemEnhancedRequest](#) memiliki `ignoreNulls()` metode yang memungkinkan Anda mengonfigurasi apa yang dilakukan SDK dengan null nilai pada objek model. Jika `ignoreNulls()` metode mengembalikan true, SDK tidak menghapus nilai atribut tabel untuk atribut objek data yang null. Jika `ignoreNulls()` metode mengembalikan false, SDK meminta layanan DynamoDB untuk menghapus atribut dari item dalam tabel. Nilai default untuk `ignoreNulls` adalah false.

Pernyataan setelah baris komentar 4 menunjukkan variasi permintaan hapus yang mengambil objek data. Klien yang disempurnakan mengekstrak nilai-nilai kunci sebelum mengirimkan permintaan akhir.

```
public static void transactWriteItems(DynamoDbEnhancedClient enhancedClient,
                                     DynamoDbTable<ProductCatalog> catalogTable,
                                     DynamoDbTable<MovieActor> movieActorTable) {

    enhancedClient.transactWriteItems(b -> b
        // 1. Simplest variation of put item request.
        .addPutItem(catalogTable, getProductCatId2())
```

```

        // 2. Put item request variation that accommodates condition
expressions.
        .addPutItem(movieActorTable,
TransactPutItemEnhancedRequest.builder(MovieActor.class)
            .item(getMovieActorStreep())

        .conditionExpression(Expression.builder().expression("attribute_not_exists
(movie)").build())
            .build())
        // 3. Update request that does not remove attribute values on the table
if the data object's value is null.
        .addUpdateItem(catalogTable,
TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
            .item(getProductCatId4ForUpdate())
            .ignoreNulls(Boolean.TRUE)
            .build())
        // 4. Variation of delete request that accepts a data object. The key
values are extracted for the request.
        .addDeleteItem(movieActorTable, getMovieActorBlanchett())
    );
}

```

Metode pembantu berikut menyediakan objek data untuk add\*Item parameter.

Metode pembantu

```

public static ProductCatalog getProductCatId2() {
    return ProductCatalog.builder()
        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
}

public static ProductCatalog getProductCatId4ForUpdate() {
    return ProductCatalog.builder()
        .id(4)
        .price(BigDecimal.valueOf(40.00))
        .title("Title 1")
        .build();
}

```

```
public static MovieActor getMovieActorBlanchett() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Cate Blanchett");
    movieActor.setMovieName("Tar");
    movieActor.setActingYear(2022);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("National Institute of Dramatic Art");
    return movieActor;
}

public static MovieActor getMovieActorStreep() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Meryl Streep");
    movieActor.setMovieName("Sophie's Choice");
    movieActor.setActingYear(1982);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("Yale School of Drama");
    return movieActor;
}
```

Tabel DynamoDB berisi item berikut sebelum contoh kode berjalan.

```
1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress',
  actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}
```

Item berikut ada di tabel setelah kode selesai berjalan.

```
3 | ProductCatalog{id=2, title='Title 55', isbn='1-565-85698', authors=[a, b],
  price=30.22}
4 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=40.0}
5 | MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
  Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
```

Item pada baris 2 telah dihapus dan baris 3 dan 5 menunjukkan item yang diletakkan. Baris 4 menunjukkan pembaruan baris 1. `price` Nilai adalah satu-satunya nilai yang berubah pada item. Jika `ignoreNulls()` telah mengembalikan `false`, baris 4 akan terlihat seperti baris berikut.

```
ProductCatalog{id=4, title='Title 1', isbn='null', authors=null, price=40.0}
```

## Contoh pemeriksaan kondisi

Contoh berikut menunjukkan penggunaan pemeriksaan kondisi. Pemeriksaan kondisi digunakan untuk memeriksa apakah ada item atau untuk memeriksa kondisi atribut tertentu dari suatu item dalam database. Item yang dicek dalam pemeriksaan kondisi tidak dapat digunakan dalam operasi lain dalam transaksi.

### Note

Anda tidak dapat menargetkan item yang sama dengan beberapa operasi dalam transaksi yang sama. Misalnya, Anda tidak dapat melakukan pemeriksaan kondisi dan juga mencoba memperbarui item yang sama dalam transaksi yang sama.

Contoh menunjukkan salah satu dari setiap jenis operasi dalam permintaan item tulis transaksional. Setelah baris komentar 2, `addConditionCheck()` metode memasok kondisi yang gagal transaksi jika `conditionExpression` parameter mengevaluasi `false`. Ekspresi kondisi yang dikembalikan dari metode yang ditampilkan dalam blok Metode Pembantu memeriksa apakah tahun penghargaan untuk film *Sophie's Choice* tidak sama dengan 1982. Jika ya, ekspresi mengevaluasi `false` dan transaksi gagal.

Panduan ini membahas [ekspresi](#) secara mendalam dalam topik lain.

```
public static void conditionCheckFailExample(DynamoDbEnhancedClient enhancedClient,
                                             DynamoDbTable<ProductCatalog>
catalogTable,
                                             DynamoDbTable<MovieActor>
movieActorTable) {
    try {
        enhancedClient.transactWriteItems(b -> b
            // 1. Perform one of each type of operation with the next three
            methods.
                .addPutItem(catalogTable,
                    TransactPutItemEnhancedRequest.builder(ProductCatalog.class)
                        .item(getProductCatId2()).build())
                .addUpdateItem(catalogTable,
                    TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
                        .item(getProductCatId4ForUpdate())
                        .ignoreNulls(Boolean.TRUE).build())
```

```

        .addDeleteItem(movieActorTable,
TransactDeleteItemEnhancedRequest.builder()
            .key(b1 -> b1

.partitionValue(getMovieActorBlanchett().getMovieName())

.sortValue(getMovieActorBlanchett().getActorName())).build()
    // 2. Add a condition check on a table item that is not involved in
another operation in this request.
        .addConditionCheck(movieActorTable, ConditionCheck.builder()
            .conditionExpression(buildConditionCheckExpression())
            .key(k -> k
                .partitionValue("Sophie's Choice")
                .sortValue("Meryl Streep"))
    // 3. Specify the request to return existing values from
the item if the condition evaluates to true.
        .returnValuesOnConditionCheckFailure(ReturnValuesOnConditionCheckFailure.ALL_OLD)
            .build())
            .build());
    // 4. Catch the exception if the transaction fails and log the information.
} catch (TransactionCanceledException ex) {
    ex.cancellationReasons().stream().forEach(cancellationReason -> {
        logger.info(cancellationReason.toString());
    });
}
}
}

```

Metode pembantu berikut digunakan dalam contoh kode sebelumnya.

Metode pembantu

```

private static Expression buildConditionCheckExpression() {
    Map<String, AttributeValue> expressionValue = Map.of(
        ":year", numberValue(1982));

    return Expression.builder()
        .expression("actingyear <> :year")
        .expressionValues(expressionValue)
        .build();
}

public static ProductCatalog getProductCatId2() {
    return ProductCatalog.builder()

```



```

        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
    }

    public static ProductCatalog getProductCatId4ForUpdate() {
        return ProductCatalog.builder()
            .id(4)
            .price(BigDecimal.valueOf(40.00))
            .title("Title 1")
            .build();
    }

    public static MovieActor getMovieActorBlanchett() {
        MovieActor movieActor = new MovieActor();
        movieActor.setActorName("Cate Blanchett");
        movieActor.setMovieName("Blue Jasmine");
        movieActor.setActingYear(2013);
        movieActor.setActingAward("Best Actress");
        movieActor.setActingSchoolName("National Institute of Dramatic Art");
        return movieActor;
    }
}

```

Tabel DynamoDB berisi item berikut sebelum contoh kode berjalan.

```

1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
3 | MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}

```

Item berikut ada di tabel setelah kode selesai berjalan.

```

ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}

```

Item tetap tidak berubah dalam tabel karena transaksi gagal. `actingYear` Nilai untuk film `Sophie's Choice` adalah 1982, seperti yang ditunjukkan pada baris 2 item dalam tabel sebelum `transactWriteItem()` metode dipanggil.

Untuk menangkap informasi pembatalan untuk transaksi, lampirkan panggilan `transactWriteItems()` metode dalam `try` blok `dancatch`.

[TransactionCanceledException](#) Setelah baris komentar 4 dari contoh, kode mencatat setiap [CancellationReason](#) objek. Karena kode berikut baris komentar 3 dari contoh menentukan bahwa nilai harus dikembalikan untuk item yang menyebabkan transaksi gagal, log menampilkan nilai database mentah untuk item `Sophie's Choice` film.

```
CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Item={actor=AttributeValue(S=Meryl Streep),
  movie=AttributeValue(S=Sophie's Choice), actingaward=AttributeValue(S=Best Actress),
  actingyear=AttributeValue(N=1982), actingschoolname=AttributeValue(S=Yale School of
  Drama)}, -
  Code=ConditionalCheckFailed, Message=The conditional request failed.)
```

### Contoh kondisi operasi tunggal

Contoh berikut menunjukkan penggunaan kondisi pada operasi tunggal dalam permintaan transaksi. Operasi hapus setelah baris komentar 1 berisi kondisi yang memeriksa nilai item target operasi terhadap database. Dalam contoh ini, ekspresi kondisi yang dibuat dengan metode pembantu setelah baris komentar 2 menentukan bahwa item harus dihapus dari database jika tahun akting film tidak sama dengan 2013.

[Ekspresi](#) dibahas nanti dalam panduan ini.

```
public static void singleOperationConditionFailExample(DynamoDbEnhancedClient
enhancedClient,
DynamoDbTable<ProductCatalog> catalogTable,
DynamoDbTable<MovieActor>
movieActorTable) {
    try {
        enhancedClient.transactWriteItems(b -> b
            .addPutItem(catalogTable,
                TransactPutItemEnhancedRequest.builder(ProductCatalog.class)
                    .item(getProductCatId2()))
```

```

        .build())
        .addUpdateItem(catalogTable,
TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
        .item(getProductCatId4ForUpdate())
        .ignoreNulls(Boolean.TRUE).build())
        // 1. Delete operation that contains a condition expression
        .addDeleteItem(movieActorTable,
TransactDeleteItemEnhancedRequest.builder()
        .key((Key.Builder k) -> {
            MovieActor blanchett = getMovieActorBlanchett();
            k.partitionValue(blanchett.getMovieName())
                .sortValue(blanchett.getActorName());
        })
        .conditionExpression(buildDeleteItemExpression()))

    .returnValuesOnConditionCheckFailure(ReturnValuesOnConditionCheckFailure.ALL_OLD)
        .build())
        .build());
    } catch (TransactionCanceledException ex) {
        ex.cancellationReasons().forEach(cancellationReason ->
logger.info(cancellationReason.toString()));
    }
}

// 2. Provide condition expression to check if 'actingyear' is not equal to 2013.
private static Expression buildDeleteItemExpression() {
    Map<String, AttributeValue> expressionValue = Map.of(
        ":year", numberValue(2013));

    return Expression.builder()
        .expression("actingyear <> :year")
        .expressionValues(expressionValue)
        .build();
}
}

```

Metode pembantu berikut digunakan dalam contoh kode sebelumnya.

Metode pembantu

```

public static ProductCatalog getProductCatId2() {
    return ProductCatalog.builder()
        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))

```

```

        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
    }

    public static ProductCatalog getProductCatId4ForUpdate() {
        return ProductCatalog.builder()
            .id(4)
            .price(BigDecimal.valueOf(40.00))
            .title("Title 1")
            .build();
    }

    public static MovieActor getMovieActorBlanchett() {
        MovieActor movieActor = new MovieActor();
        movieActor.setActorName("Cate Blanchett");
        movieActor.setMovieName("Blue Jasmine");
        movieActor.setActingYear(2013);
        movieActor.setActingAward("Best Actress");
        movieActor.setActingSchoolName("National Institute of Dramatic Art");
        return movieActor;
    }
}

```

Tabel DynamoDB berisi item berikut sebelum contoh kode berjalan.

```

1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}

```

Item berikut ada di tabel setelah kode selesai berjalan.

```

ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2023-03-15 11:29:07 [main] INFO org.example.tests.TransactDemoTest:168 -
MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}

```

Item tetap tidak berubah dalam tabel karena transaksi gagal. `actingYear` nilai untuk film Blue Jasmine adalah 2013 seperti yang ditunjukkan pada baris 2 dalam daftar item sebelum contoh kode berjalan.

Baris berikut dicatat ke konsol.

```

CancellationReason(Code=None)

```

```

CancellationReason(Code=None)
CancellationReason(Item={actor=AttributeValue(S=Cate Blanchett),
  movie=AttributeValue(S=Blue Jasmine), actingaward=AttributeValue(S=Best Actress),
  actingyear=AttributeValue(N=2013), actingschoolname=AttributeValue(S=National
  Institute of Dramatic Art)},
  Code=ConditionalCheckFailed, Message=The conditional request failed)

```

## Gunakan indeks sekunder

Indeks sekunder meningkatkan akses data dengan mendefinisikan kunci alternatif yang Anda gunakan dalam operasi kueri dan pemindaian. Indeks sekunder global (GSI) memiliki kunci partisi dan kunci pengurutan yang dapat berbeda dari yang ada di tabel dasar. Sebaliknya, indeks sekunder lokal (LSI) menggunakan kunci partisi dari indeks primer.

Anotasi kelas data dengan anotasi indeks sekunder

Atribut yang berpartisipasi dalam indeks sekunder memerlukan `@DynamoDbSecondarySortKey` anotasi `@DynamoDbSecondaryPartitionKey` atau anotasi.

Kelas berikut menunjukkan anotasi untuk dua indeks. GSI bernama `SubjectLastPostedDateIndex` menggunakan `Subject` atribut untuk kunci partisi dan `LastPostedDateTime` untuk kunci sortir. LSI bernama `ForumLastPostedDateIndex` menggunakan `ForumName` sebagai kunci partisi dan `LastPostedDateTime` sebagai kunci sortir.

Perhatikan bahwa `Subject` atribut tersebut memiliki peran ganda. Ini adalah kunci sortir kunci primer dan kunci partisi dari GSI bernama `SubjectLastPostedDateIndex`.

## MessageThread kelas

`MessageThread` kelas ini cocok untuk digunakan sebagai kelas data untuk [tabel Thread contoh](#) di Amazon DynamoDB Developer Guide.

Impor

```

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import
  software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
  software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import
  software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

```

```
import java.util.List;
```

```
@DynamoDbBean
public class MessageThread {
    private String ForumName;
    private String Subject;
    private String Message;
    private String LastPostedBy;
    private String LastPostedDateTime;
    private Integer Views;
    private Integer Replies;
    private Integer Answered;
    private List<String> Tags;

    @DynamoDbPartitionKey
    public String getForumName() {
        return ForumName;
    }

    public void setForumName(String forumName) {
        ForumName = forumName;
    }

    // Sort key for primary index and partition key for GSI
    "SubjectLastPostedDateIndex".
    @DynamoDbSortKey
    @DynamoDbSecondaryPartitionKey(indexNames = "SubjectLastPostedDateIndex")
    public String getSubject() {
        return Subject;
    }

    public void setSubject(String subject) {
        Subject = subject;
    }

    // Sort key for GSI "SubjectLastPostedDateIndex" and sort key for LSI
    "ForumLastPostedDateIndex".
    @DynamoDbSecondarySortKey(indexNames = {"SubjectLastPostedDateIndex",
    "ForumLastPostedDateIndex"})
    public String getLastPostedDateTime() {
        return LastPostedDateTime;
    }
}
```

```
public void setLastPostedDateTime(String lastPostedDateTime) {
    LastPostedDateTime = lastPostedDateTime;
}
public String getMessage() {
    return Message;
}

public void setMessage(String message) {
    Message = message;
}

public String getLastPostedBy() {
    return LastPostedBy;
}

public void setLastPostedBy(String lastPostedBy) {
    LastPostedBy = lastPostedBy;
}

public Integer getViews() {
    return Views;
}

public void setViews(Integer views) {
    Views = views;
}

@DynamoDbSecondaryPartitionKey(indexNames = "ForumRepliesIndex")
public Integer getReplies() {
    return Replies;
}

public void setReplies(Integer replies) {
    Replies = replies;
}

public Integer getAnswered() {
    return Answered;
}

public void setAnswered(Integer answered) {
    Answered = answered;
}
```

```
public List<String> getTags() {
    return Tags;
}

public void setTags(List<String> tags) {
    Tags = tags;
}

public MessageThread() {
    this.Answered = 0;
    this.LastPostedBy = "";
    this.ForumName = "";
    this.Message = "";
    this.LastPostedDateTime = "";
    this.Replies = 0;
    this.Views = 0;
    this.Subject = "";
}

@Override
public String toString() {
    return "MessageThread{" +
        "ForumName='" + ForumName + '\'' +
        ", Subject='" + Subject + '\'' +
        ", Message='" + Message + '\'' +
        ", LastPostedBy='" + LastPostedBy + '\'' +
        ", LastPostedDateTime='" + LastPostedDateTime + '\'' +
        ", Views=" + Views +
        ", Replies=" + Replies +
        ", Answered=" + Answered +
        ", Tags=" + Tags +
        '}';
}
}
```

## Buat indeks

Dimulai dengan versi 2.20.86 SDK for Java, `createTable()` metode ini secara otomatis menghasilkan indeks sekunder dari anotasi kelas data. Secara default, semua atribut dari tabel dasar disalin ke indeks dan nilai throughput yang disediakan adalah 20 unit kapasitas baca dan 20 unit kapasitas tulis.



Namun, jika Anda menggunakan versi SDK sebelum 2.20.86, Anda perlu membuat indeks bersama dengan tabel seperti yang ditunjukkan pada contoh berikut. Contoh ini membangun dua indeks untuk tabel. Thread Parameter [builder](#) memiliki metode untuk mengkonfigurasi kedua jenis indeks seperti yang ditunjukkan setelah baris komentar 1 dan 2. Anda menggunakan `indexName()` metode pembuat indeks untuk mengaitkan nama indeks yang ditentukan dalam anotasi kelas data dengan jenis indeks yang dimaksud.

Kode ini mengkonfigurasi semua atribut tabel untuk berakhir di kedua indeks setelah baris komentar 3 dan 4. Informasi selengkapnya tentang [proyeksi atribut](#) tersedia di Panduan Pengembang Amazon DynamoDB.

```
public static void createMessageThreadTable(DynamoDbTable<MessageThread>
messageThreadDynamoDbTable, DynamoDbClient dynamoDbClient) {
    messageThreadDynamoDbTable.createTable(b -> b
        // 1. Generate the GSI.
        .globalSecondaryIndices(gsi ->
gsi.indexName("SubjectLastPostedDateIndex")
            // 3. Populate the GSI with all attributes.
            .projection(p -> p
                .projectionType(ProjectionType.ALL))
        )
        // 2. Generate the LSI.
        .localSecondaryIndices(lsi -> lsi.indexName("ForumLastPostedDateIndex")
            // 4. Populate the LSI with all attributes.
            .projection(p -> p
                .projectionType(ProjectionType.ALL))
        )
    );
}
```

Kueri dengan menggunakan indeks

Contoh berikut query indeks `ForumLastPostedDateIndex` sekunder lokal.

Mengikuti baris komentar 2, Anda membuat [QueryConditional](#) objek yang diperlukan saat memanggil [DynamoDbIndexmetode.query\(\)](#).

Anda mendapatkan referensi ke indeks yang ingin Anda kueri setelah baris komentar 3 dengan meneruskan nama indeks. Mengikuti baris komentar 4, Anda memanggil `query()` metode pada indeks yang lewat di `QueryConditional` objek.

Anda juga mengonfigurasi kueri untuk mengembalikan tiga nilai atribut seperti yang ditunjukkan setelah baris komentar 5. Jika tidak `attributesToProject()` dipanggil, query mengembalikan

semua nilai atribut. Perhatikan bahwa nama atribut yang ditentukan dimulai dengan huruf kecil. Nama atribut ini cocok dengan yang digunakan dalam tabel, belum tentu nama atribut dari kelas data.

Mengikuti baris komentar 6, ulangi hasil dan log setiap item yang dikembalikan oleh kueri dan juga menyimpannya dalam daftar untuk kembali ke pemanggil.

```
public static List<MessageThread> queryUsingSecondaryIndices(DynamoDbEnhancedClient
    enhancedClient,
                                                            String lastPostedDate,
                                                            DynamoDbTable<MessageThread> threadTable) {
    // 1. Log the parameter value.
    logger.info("lastPostedDate value: {}", lastPostedDate);

    // 2. Create a QueryConditional whose sort key value must be greater than or
    equal to the parameter value.
    QueryConditional queryConditional =
    QueryConditional.sortGreaterThanOrEqualTo(qc ->
        qc.partitionValue("Forum02").sortValue(lastPostedDate));

    // 3. Specify the index name to query the DynamoDbIndex instance.
    final DynamoDbIndex<MessageThread> forumLastPostedDateIndex =
    threadTable.index("ForumLastPostedDateIndex");

    // 4. Perform the query by using the QueryConditional object.
    final SdkIterable<Page<MessageThread>> pagedResult =
    forumLastPostedDateIndex.query(q -> q
        .queryConditional(queryConditional)
        // 5. Request three attribute in the results.
        .attributesToProject("forumName", "subject", "lastPostedDateTime"));

    List<MessageThread> collectedItems = new ArrayList<>();
    // 6. Iterate through the pages response and sort the items.
    pagedResult.stream().forEach(page -> page.items().stream()

    .sorted(Comparator.comparing(MessageThread::getLastPostedDateTime))
        .forEach(mt -> {
            // 7. Log the returned items and add the collection to
            return to the caller.
            logger.info(mt.toString());
            collectedItems.add(mt);
        }));
    return collectedItems;
}
```

```
}
```

Item berikut ada dalam database sebelum query dijalankan.

```
MessageThread{ForumName='Forum01', Subject='Subject01', Message='Message01',  
  LastPostedBy='', LastPostedDateTime='2023.03.28', Views=0, Replies=0, Answered=0,  
  Tags=null}  
MessageThread{ForumName='Forum02', Subject='Subject02', Message='Message02',  
  LastPostedBy='', LastPostedDateTime='2023.03.29', Views=0, Replies=0, Answered=0,  
  Tags=null}  
MessageThread{ForumName='Forum02', Subject='Subject04', Message='Message04',  
  LastPostedBy='', LastPostedDateTime='2023.03.31', Views=0, Replies=0, Answered=0,  
  Tags=null}  
MessageThread{ForumName='Forum02', Subject='Subject08', Message='Message08',  
  LastPostedBy='', LastPostedDateTime='2023.04.04', Views=0, Replies=0, Answered=0,  
  Tags=null}  
MessageThread{ForumName='Forum02', Subject='Subject10', Message='Message10',  
  LastPostedBy='', LastPostedDateTime='2023.04.06', Views=0, Replies=0, Answered=0,  
  Tags=null}  
MessageThread{ForumName='Forum03', Subject='Subject03', Message='Message03',  
  LastPostedBy='', LastPostedDateTime='2023.03.30', Views=0, Replies=0, Answered=0,  
  Tags=null}  
MessageThread{ForumName='Forum03', Subject='Subject06', Message='Message06',  
  LastPostedBy='', LastPostedDateTime='2023.04.02', Views=0, Replies=0, Answered=0,  
  Tags=null}  
MessageThread{ForumName='Forum03', Subject='Subject09', Message='Message09',  
  LastPostedBy='', LastPostedDateTime='2023.04.05', Views=0, Replies=0, Answered=0,  
  Tags=null}  
MessageThread{ForumName='Forum05', Subject='Subject05', Message='Message05',  
  LastPostedBy='', LastPostedDateTime='2023.04.01', Views=0, Replies=0, Answered=0,  
  Tags=null}  
MessageThread{ForumName='Forum07', Subject='Subject07', Message='Message07',  
  LastPostedBy='', LastPostedDateTime='2023.04.03', Views=0, Replies=0, Answered=0,  
  Tags=null}
```

Pernyataan logging pada baris 1 dan 6 menghasilkan output konsol berikut.

```
lastPostedDate value: 2023.03.31  
MessageThread{ForumName='Forum02', Subject='Subject04', Message='', LastPostedBy='',  
  LastPostedDateTime='2023.03.31', Views=0, Replies=0, Answered=0, Tags=null}  
MessageThread{ForumName='Forum02', Subject='Subject08', Message='', LastPostedBy='',  
  LastPostedDateTime='2023.04.04', Views=0, Replies=0, Answered=0, Tags=null}
```

```
MessageThread{ForumName='Forum02', Subject='Subject10', Message='', LastPostedBy='',  
LastPostedDateTime='2023.04.06', Views=0, Replies=0, Answered=0, Tags=null}
```

Kueri mengembalikan item dengan *forumName* nilai Forum02 dan *lastPostedDateTime* nilai lebih besar dari atau sama dengan 2023.03.31. Hasilnya menunjukkan message nilai dengan string kosong meskipun message atribut memiliki nilai dalam indeks. Ini karena atribut pesan tidak diproyeksikan setelah baris komentar 5.

## Gunakan fitur pemetaan tingkat lanjut

Pelajari tentang fitur skema tabel lanjutan di DynamoDB Enhanced Client API.

Memahami jenis skema tabel

[TableSchema](#) adalah antarmuka ke fungsionalitas pemetaan DynamoDB Enhanced Client API. Hal ini dapat memetakan objek data ke dan dari peta [AttributeValues](#). Sebuah `TableSchema` objek perlu tahu tentang struktur tabel yang dipetakannya. Informasi struktur ini disimpan dalam suatu [TableMetadata](#) objek.

API klien yang disempurnakan memiliki beberapa implementasi `TableSchema`, yang mengikuti.

Skema tabel yang dihasilkan dari kelas beranotasi

Ini adalah operasi yang cukup mahal untuk membangun `TableSchema` dari kelas beranotasi, jadi kami sarankan melakukan ini sekali, saat startup aplikasi.

### [BeanTableSchema](#)

Implementasi ini dibangun berdasarkan atribut dan anotasi kelas kacang. Contoh pendekatan ini ditunjukkan di [bagian Memulai](#).

#### Note

Jika a `BeanTableSchema` tidak berperilaku seperti yang Anda harapkan, aktifkan pencatatan debug untuk `software.amazon.awssdk.enhanced.dynamodb.beans`

### [ImmutableTableSchema](#)

Implementasi ini dibangun dari kelas data yang tidak dapat diubah. Pendekatan ini dijelaskan di [??? bagian ini](#).

## Skema tabel yang dihasilkan dengan pembangun

Berikut `TableSchema` s dibangun dari kode dengan menggunakan pembangun. Pendekatan ini lebih murah daripada pendekatan yang menggunakan kelas data beranotasi. Pendekatan builder menghindari penggunaan anotasi dan tidak memerlukan standar `JavaBean` penamaan.

### [StaticTableSchema](#)

Implementasi ini dibangun untuk kelas data yang bisa berubah. Bagian memulai panduan ini menunjukkan cara [membuat StaticTableSchema menggunakan pembangun](#).

### [StaticImmutableTableSchema](#)

Demikian pula dengan cara Anda membangun `StaticTableSchema`, Anda menghasilkan implementasi jenis ini `TableSchema` menggunakan [pembangun](#) untuk digunakan dengan kelas data yang tidak dapat diubah.

## Skema tabel untuk data tanpa skema tetap

### [DocumentTableSchema](#)

Tidak seperti implementasi lain dari `TableSchema`, Anda tidak mendefinisikan atribut untuk sebuah `DocumentTableSchema` instance. Biasanya, Anda hanya menentukan kunci utama dan penyedia konverter atribut. Sebuah `EnhancedDocument` instance menyediakan atribut yang Anda bangun dari elemen individual atau dari string JSON.

## Sertakan atau keculikan atribut secara eksplisit

DynamoDB Enhanced Client API menawarkan anotasi untuk mengecualikan atribut kelas data agar tidak menjadi atribut pada tabel. Dengan API, Anda juga dapat menggunakan nama atribut yang berbeda dari nama atribut kelas data.

### Kecualikan atribut

Untuk mengabaikan atribut yang seharusnya tidak dipetakan ke tabel DynamoDB, tandai atribut dengan anotasi. `@DynamoDbIgnore`

```
private String internalKey;

@DynamoDbIgnore
public String getInternalKey() { return this.internalKey; }
```

```
public void setInternalKey(String internalKey) { return this.internalKey =
    internalKey;}
```

## Sertakan atribut

Untuk mengubah nama atribut yang digunakan dalam tabel DynamoDB, tandai dengan anotasi dan berikan nama `@DynamoDbAttribute` yang berbeda.

```
private String internalKey;

@DynamoDbAttribute("renamedInternalKey")
public String getInternalKey() { return this.internalKey; }
public void setInternalKey(String internalKey) { return this.internalKey =
    internalKey;}
```

## Konversi atribut kontrol

Secara default, skema tabel menyediakan konverter untuk semua tipe primitif dan banyak jenis Java umum melalui implementasi default antarmuka. [AttributeConverterProvider](#) Anda dapat mengubah keseluruhan perilaku default dengan `AttributeConverterProvider` implementasi kustom. Anda juga dapat mengubah konverter untuk satu atribut.

Untuk daftar konverter yang tersedia, lihat [AttributeConverter](#) antarmuka Java doc.

## Menyediakan penyedia konverter atribut khusus

Anda dapat memberikan satu `AttributeConverterProvider` atau satu rantai `AttributeConverterProvider` s yang dipesan melalui `@DynamoDbBean` (`converterProviders = {...}`) anotasi. Setiap kustom `AttributeConverterProvider` harus memperluas `AttributeConverterProvider` antarmuka.

Perhatikan bahwa jika Anda menyediakan rantai penyedia konverter atribut Anda sendiri, Anda akan mengganti penyedia konverter default, `DefaultAttributeConverterProvider`. Jika Anda ingin menggunakan fungsionalitas `DefaultAttributeConverterProvider`, Anda harus memasukkannya ke dalam rantai.

Dimungkinkan juga untuk membubuhi keterangan kacang dengan array kosong. `{}` Ini menonaktifkan penggunaan penyedia konverter atribut apa pun, termasuk default. Dalam hal ini semua atribut yang akan dipetakan harus memiliki konverter atribut mereka sendiri.

Cuplikan berikut menunjukkan penyedia konverter tunggal.

```
@DynamoDbBean(converterProviders = ConverterProvider1.class)
public class Customer {

}
```

Cuplikan berikut menunjukkan penggunaan rantai penyedia konverter. Karena default SDK disediakan terakhir, ia memiliki prioritas terendah.

```
@DynamoDbBean(converterProviders = {
    ConverterProvider1.class,
    ConverterProvider2.class,
    DefaultAttributeConverterProvider.class})
public class Customer {

}
```

Pembangun skema tabel statis memiliki `attributeConverterProviders()` metode yang bekerja dengan cara yang sama. Ini ditunjukkan dalam cuplikan berikut.

```
private static final StaticTableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
    StaticTableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("name")
            a.getter(Customer::getName)
            a.setter(Customer::setName))
        .attributeConverterProviders(converterProvider1, converterProvider2)
        .build();
```

## Ganti pemetaan atribut tunggal

Untuk mengganti cara atribut tunggal dipetakan, berikan atribut `AttributeConverter` untuk atribut. Penambahan ini mengesampingkan konverter apa pun yang disediakan oleh skema `AttributeConverterProviders` tabel. Ini menambahkan konverter khusus hanya untuk atribut itu. Atribut lain, bahkan yang dari tipe yang sama, tidak akan menggunakan konverter itu kecuali jika secara eksplisit ditentukan untuk atribut lainnya.

`@DynamoDbConvertedBy` anotasi ini digunakan untuk menentukan `AttributeConverter` kelas kustom seperti yang ditunjukkan dalam cuplikan berikut.

```
@DynamoDbBean
```

```
public class Customer {
    private String name;

    @DynamoDbConvertedBy(CustomAttributeConverter.class)
    public String getName() { return this.name; }
    public void setName(String name) { this.name = name;}
}
```

Pembangun untuk skema statis memiliki `attributeConverter()` metode pembangun atribut yang setara. Metode ini mengambil contoh dari `AttributeConverter` sebagai berikut menunjukkan.

```
private static final StaticTableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
    StaticTableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("name")
            a.getter(Customer::getName)
            a.setter(Customer::setName)
            a.attributeConverter(customAttributeConverter))
        .build();
```

## Contoh

Contoh ini menunjukkan `AttributeConverterProvider` implementasi yang menyediakan konverter atribut untuk [java.net.HttpCookie](http://java.net/HttpCookie) objek.

`SimpleUserKelas` berikut berisi atribut bernama `lastUsedCookie` yang merupakan instance dari `HttpCookie`.

Parameter untuk `@DynamoDbBean` anotasi mencantumkan dua `AttributeConverterProvider` kelas yang menyediakan konverter.

## Class with annotations

```
@DynamoDbBean(converterProviders = {CookieConverterProvider.class,
DefaultAttributeConverterProvider.class})
public static final class SimpleUser {
    private String name;
    private HttpCookie lastUsedCookie;

    @DynamoDbPartitionKey
    public String getName() {
        return name;
    }
}
```



```

    }

    public void setName(String name) {
        this.name = name;
    }

    public HttpCookie getLastUsedCookie() {
        return lastUsedCookie;
    }

    public void setLastUsedCookie(HttpCookie lastUsedCookie) {
        this.lastUsedCookie = lastUsedCookie;
    }
}

```

## Static table schema

```

private static final TableSchema<SimpleUser> SIMPLE_USER_TABLE_SCHEMA =
    TableSchema.builder(SimpleUser.class)
        .newItemSupplier(SimpleUser::new)
        .attributeConverterProviders(CookieConverterProvider.create(),
AttributeConverterProvider.defaultProvider())
        .addAttribute(String.class, a -> a.name("name")
            .setter(SimpleUser::setName)
            .getter(SimpleUser::getName)
            .tags(StaticAttributeTags.primaryPartitionKey()))
        .addAttribute(HttpCookie.class, a -> a.name("lastUsedCookie")
            .setter(SimpleUser::setLastUsedCookie)
            .getter(SimpleUser::getLastUsedCookie))
        .build();

```

CookieConverterProviderDalam contoh berikut memberikan contoh dari sebuahHttpCookeConverter.

```

public static final class CookieConverterProvider implements
AttributeConverterProvider {
    private final Map<EnhancedType<?>, AttributeConverter<?>> converterCache =
ImmutableMap.of(
        // 1. Add HttpCookieConverter to the internal cache.
        EnhancedType.of(HttpCookie.class), new HttpCookieConverter());

    public static CookieConverterProvider create() {
        return new CookieConverterProvider();
    }
}

```

```

    }

    // The SDK calls this method to find out if the provider contains a
    AttributeConverter instance
    // for the EnhancedType<T> argument.
    @SuppressWarnings("unchecked")
    @Override
    public <T> AttributeConverter<T> converterFor(EnhancedType<T> enhancedType) {
        return (AttributeConverter<T>) converterCache.get(enhancedType);
    }
}

```

## Kode konversi

Dalam `transformFrom()` metode `HttpCookieConverter` kelas berikut, kode menerima `HttpCookie` instance dan mengubahnya menjadi peta `DynamoDB` yang disimpan sebagai atribut.

`transformTo()` Metode menerima parameter peta `DynamoDB`, kemudian memanggil konstruktor `HttpCookie` yang membutuhkan nama dan nilai.

```

public static final class HttpCookieConverter implements
AttributeConverter<HttpCookie> {

    @Override
    public AttributeValue transformFrom(HttpCookie httpCookie) {

        return AttributeValue.fromM(
            Map.of ("cookieName", AttributeValue.fromS(httpCookie.getName()),
                "cookieValue", AttributeValue.fromS(httpCookie.getValue()))
        );
    }

    @Override
    public HttpCookie transformTo(AttributeValue attributeValue) {
        Map<String, AttributeValue> map = attributeValue.m();
        return new HttpCookie(
            map.get("cookieName").s(),
            map.get("cookieValue").s());
    }

    @Override
    public EnhancedType<HttpCookie> type() {
        return EnhancedType.of(HttpCookie.class);
    }
}

```

```

    }

    @Override
    public AttributeValueType attributeValueType() {
        return AttributeValueType.M;
    }
}

```

## Ubah perilaku pembaruan atribut

Anda dapat menyesuaikan perilaku pembaruan atribut individual saat Anda melakukan operasi pembaruan. [Beberapa contoh operasi pembaruan di DynamoDB Enhanced Client API adalah `updateItem\(\)` dan `transactWriteItems\(\)`](#).

Misalnya, bayangkan Anda ingin menyimpan stempel waktu yang dibuat pada catatan Anda. Namun, Anda ingin nilainya ditulis hanya jika tidak ada nilai yang ada untuk atribut yang sudah ada dalam database. Dalam hal ini, Anda menggunakan perilaku [WRITE\\_IF\\_NOT\\_EXISTS](#) pembaruan.

Contoh berikut menunjukkan anotasi yang menambahkan perilaku ke `createdOn` atribut.

```

@DynamoDbBean
public class Customer extends GenericRecord {
    private String id;
    private Instant createdOn;

    @DynamoDbPartitionKey
    public String getId() { return this.id; }
    public void setId(String id) { this.name = id; }

    @DynamoDbUpdateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS)
    public Instant getCreatedOn() { return this.createdOn; }
    public void setCreatedOn(Instant createdOn) { this.createdOn = createdOn; }
}

```

Anda dapat mendeklarasikan perilaku pembaruan yang sama ketika Anda membangun skema tabel statis seperti yang ditunjukkan pada contoh berikut setelah baris komentar 1.

```

static final TableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
    TableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("id")
            .getter(Customer::getId)

```

```

        .setter(Customer::setId)

.tags(StaticAttributeTags.primaryPartitionKey())
    .addAttribute(Instant.class, a -> a.name("createdOn")
        .getter(Customer::getCreatedOn)
        .setter(Customer::setCreatedOn)
        // 1. Add an UpdateBehavior.

.tags(StaticAttributeTags.updateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS))
    .build();

```

## Ratakan atribut dari kelas lain

Jika atribut untuk tabel Anda tersebar di beberapa kelas Java yang berbeda, baik melalui pewarisan atau komposisi, DynamoDB Enhanced Client API menyediakan dukungan untuk meratakan atribut menjadi satu kelas.

## Gunakan warisan

Jika kelas Anda menggunakan pewarisan, gunakan pendekatan berikut untuk meratakan hierarki.

## Gunakan kacang berannotasi

Untuk pendekatan anotasi, kedua kelas harus membawa `@DynamoDbBean` anotasi dan kelas harus membawa satu atau lebih anotasi kunci primer.

Berikut ini menunjukkan contoh kelas data yang memiliki hubungan warisan.

## Standard data class

```

@DynamoDbBean
public class Customer extends GenericRecord {
    private String name;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}

@DynamoDbBean
public abstract class GenericRecord {
    private String id;
    private String createDate;
}

```

```

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedDate() { return createdDate; }
    public void setCreatedDate(String createdDate) { this.createdDate =
createdDate; }
}

```

## Lombok

[onMethodOpsi](#) Lombok menyalin anotasi DynamoDB berbasis atribut, seperti, ke kode yang dihasilkan. `@DynamoDbPartitionKey`

```

@DynamoDbBean
@Data
@ToString(callSuper = true)
public class Customer extends GenericRecord {
    private String name;
}

@Data
@DynamoDbBean
public abstract class GenericRecord {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;
    private String createdDate;
}

```

## Gunakan skema statis

Untuk pendekatan skema statis, gunakan `extend()` metode pembangun untuk menciutkan atribut kelas induk ke kelas anak. Ini ditampilkan setelah komentar baris 1 dalam contoh berikut.

```

    StaticTableSchema<org.example.tests.model.inheritance.stat.GenericRecord>
    GENERIC_RECORD_SCHEMA =

    StaticTableSchema.builder(org.example.tests.model.inheritance.stat.GenericRecord.class)
        // The partition key will be inherited by the top level mapper.
        .addAttribute(String.class, a -> a.name("id"))

    .getter(org.example.tests.model.inheritance.stat.GenericRecord::getId)

```

```

.setter(org.example.tests.model.inheritance.stat.GenericRecord::setId)
        .tags(primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("created_date"))

.getter(org.example.tests.model.inheritance.stat.GenericRecord::getCreatedDate)

.setter(org.example.tests.model.inheritance.stat.GenericRecord::setCreatedDate))
        .build();

    StaticTableSchema<org.example.tests.model.inheritance.stat.Customer>
CUSTOMER_SCHEMA =

StaticTableSchema.builder(org.example.tests.model.inheritance.stat.Customer.class)

.newItemSupplier(org.example.tests.model.inheritance.stat.Customer::new)
        .addAttribute(String.class, a -> a.name("name"))

.getter(org.example.tests.model.inheritance.stat.Customer::getName)

.setter(org.example.tests.model.inheritance.stat.Customer::setName))
        // 1. Use the extend() method to collapse the parent attributes
onto the child class.
        .extend(GENERIC_RECORD_SCHEMA) // All the attributes of the
GenericRecord schema are added to Customer.
        .build();

```

Contoh skema statis sebelumnya menggunakan kelas data berikut. Karena pemetaan didefinisikan ketika Anda membangun skema tabel statis, kelas data tidak memerlukan anotasi.

## Kelas data

### Standard data class

```

public class Customer extends GenericRecord {
    private String name;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}

public abstract class GenericRecord {

```

```
private String id;
private String createdAt;

public String getId() { return id; }
public void setId(String id) { this.id = id; }

public String getCreatedAt() { return createdAt; }
public void setCreatedAt(String createdAt) { this.createdAt =
createdAt; }
```

## Lombok

```
@Data
@ToString(callSuper = true)
public class Customer extends GenericRecord{
    private String name;
}

@Data
public abstract class GenericRecord {
    private String id;
    private String createdAt;
}
```

## Gunakan komposisi

Jika kelas Anda menggunakan komposisi, gunakan pendekatan berikut untuk meratakan hierarki.

## Gunakan kacang beranotasi

`@DynamoDbFlatten` anotasi meratakan kelas yang terkandung.

Contoh kelas data berikut menggunakan `@DynamoDbFlatten` anotasi untuk secara efektif menambahkan semua atribut kelas yang terkandung ke `GenericRecord` Customer kelas.

## Standard data class

```
@DynamoDbBean
public class Customer {
    private String name;
    private GenericRecord record;
```

```

public String getName() { return this.name; }
public void setName(String name) { this.name = name; }

@DynamoDbFlatten
public GenericRecord getRecord() { return this.record; }
public void setRecord(GenericRecord record) { this.record = record; }

@DynamoDbBean
public class GenericRecord {
    private String id;
    private String createdAt;

    @DynamoDbPartitionKey
    public String getId() { return this.id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return this.createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
createdAt; }
}

```

## Lombok

```

@Data
@DynamoDbBean
public class Customer {
    private String name;
    @Getter(onMethod_=@DynamoDbFlatten)
    private GenericRecord record;
}

@Data
@DynamoDbBean
public class GenericRecord {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;
    private String createdAt;
}

```

Anda dapat menggunakan anotasi rata untuk meratakan sebanyak mungkin kelas yang memenuhi syarat yang Anda butuhkan. Pembatasan berikut berlaku:



- Semua nama atribut harus unik setelah diratakan.
- Tidak boleh ada lebih dari satu kunci partisi, kunci sortir, atau nama tabel.

## Gunakan skema statis

Saat Anda membuat skema tabel statis, gunakan `flatten()` metode pembangun. Anda juga menyediakan metode pengambil dan penyetel yang mengidentifikasi kelas yang terkandung.

```
StaticTableSchema<GenericRecord> GENERIC_RECORD_SCHEMA =
    StaticTableSchema.builder(GenericRecord.class)
        .newItemSupplier(GenericRecord::new)
        .addAttribute(String.class, a -> a.name("id")
            .getter(GenericRecord::getId)
            .setter(GenericRecord::setId)
            .tags(primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("created_date")
            .getter(GenericRecord::getCreatedDate)
            .setter(GenericRecord::setCreatedDate))
        .build();

StaticTableSchema<Customer> CUSTOMER_SCHEMA =
    StaticTableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("name")
            .getter(Customer::getName)
            .setter(Customer::setName))
        // Because we are flattening a component object, we supply a
getter and setter so the
        // mapper knows how to access it.
        .flatten(GENERIC_RECORD_SCHEMA, Customer::getRecord,
Customer::setRecord)
        .build();
```

Contoh skema statis sebelumnya menggunakan kelas data berikut.

## Kelas data

### Standard data class

```
public class Customer {
    private String name;
```

```

private GenericRecord record;

public String getName() { return this.name; }
public void setName(String name) { this.name = name; }

public GenericRecord getRecord() { return this.record; }
public void setRecord(GenericRecord record) { this.record = record; }

public class GenericRecord {
    private String id;
    private String createdAt;

    public String getId() { return this.id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return this.createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
createdAt; }
}

```

## Lombok

```

@Data
public class Customer {
    private String name;
    private GenericRecord record;
}

@Data
public class GenericRecord {
    private String id;
    private String createdAt;
}

```

Anda dapat menggunakan pola pembangun untuk meratakan sebanyak mungkin kelas yang memenuhi syarat yang Anda butuhkan.

### Implikasi untuk kode lain

Bila Anda menggunakan `@DynamoDbFlatten` atribut (atau metode `flatten()` builder), item di DynamoDB berisi atribut untuk setiap atribut dari objek yang disusun. Ini juga mencakup atribut dari objek penyusun.

Sebaliknya, jika Anda membuat anotasi kelas data dengan kelas tersusun dan tidak menggunakan `@DynamoDbFlatten`, item disimpan dengan objek tersusun sebagai atribut tunggal.

Misalnya, bandingkan `Customer` kelas yang ditunjukkan dalam [perataan dengan contoh komposisi dengan](#) dan tanpa perataan atribut. `record` Anda dapat memvisualisasikan perbedaannya dengan JSON seperti yang ditunjukkan pada tabel berikut.

Dengan perataan	Tanpa perataan
3 atribut	2 atribut
<pre>{   "id": "1",   "createdDate": "today",   "name": "my name" }</pre>	<pre>{   "id": "1",   "record": {     "createdDate": "today",     "name": "my name"   } }</pre>

Perbedaannya menjadi penting jika Anda memiliki kode lain yang mengakses tabel DynamoDB yang mengharapkan untuk menemukan atribut tertentu.

### Bekerja dengan atribut bersarang

Sebuah atribut bersarang di DynamoDB disematkan dalam atribut lain. Contohnya adalah elemen daftar dan entri peta.

Di Java, atribut bersarang DynamoDB sesuai dengan anggota kelas yang merupakan `or`. `List` `Map` Ini juga sesuai dengan contoh dari jenis kompleks, seperti `Address` atau `PhoneNumber`, seperti yang digunakan dalam `Person` kelas berikut.

### Person kelas

```
@DynamoDbBean
public class Person {
    Integer id;
    String firstName;
    String lastName;
    Integer age;
    Map<String, Address> addresses;
```

```
List<PhoneNumber> phoneNumbers;

List<String> hobbies;

@DynamoDbPartitionKey
public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public Integer getAge() {
    return age;
}

public void setAge(Integer age) {
    this.age = age;
}

public Map<String, Address> getAddresses() {
    return addresses;
}

public void setAddresses(Map<String, Address> addresses) {
    this.addresses = addresses;
}
```

```
public List<PhoneNumber> getPhoneNumbers() {
    return phoneNumbers;
}

public void setPhoneNumbers(List<PhoneNumber> phoneNumbers) {
    this.phoneNumbers = phoneNumbers;
}

public List<String> getHobbies() {
    return hobbies;
}

public void setHobbies(List<String> hobbies) {
    this.hobbies = hobbies;
}

@Override
public String toString() {
    return "Person{" +
        "id=" + id +
        ", firstName='" + firstName + '\'' +
        ", lastName='" + lastName + '\'' +
        ", age=" + age +
        ", addresses=" + addresses +
        ", phoneNumbers=" + phoneNumbers +
        ", hobbies=" + hobbies +
        '}';
}
}
```

## Address kelas

```
@DynamoDbBean
public class Address {
    private String street;
    private String city;
    private String state;
    private String zipCode;

    public Address() {
    }
}
```

```
public String getStreet() {
    return this.street;
}

public String getCity() {
    return this.city;
}

public String getState() {
    return this.state;
}

public String getZipCode() {
    return this.zipCode;
}

public void setStreet(String street) {
    this.street = street;
}

public void setCity(String city) {
    this.city = city;
}

public void setState(String state) {
    this.state = state;
}

public void setZipCode(String zipCode) {
    this.zipCode = zipCode;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Address address = (Address) o;
    return Objects.equals(street, address.street) && Objects.equals(city,
address.city) && Objects.equals(state, address.state) && Objects.equals(zipCode,
address.zipCode);
}

@Override
public int hashCode() {
```

```
        return Objects.hash(street, city, state, zipCode);
    }

    @Override
    public String toString() {
        return "Address{" +
            "street='" + street + '\'' +
            ", city='" + city + '\'' +
            ", state='" + state + '\'' +
            ", zipCode='" + zipCode + '\'' +
            '}';
    }
}
```

## PhoneNumber kelas

```
@DynamoDbBean
public class PhoneNumber {
    String type;
    String number;

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getNumber() {
        return number;
    }

    public void setNumber(String number) {
        this.number = number;
    }

    @Override
    public String toString() {
        return "PhoneNumber{" +
            "type='" + type + '\'' +
            ", number='" + number + '\'' +
            '}';
    }
}
```

```
}
}
```

Petakan atribut bersarang

Gunakan kelas beranotasi

Anda dapat menyimpan atribut bersarang untuk kelas kustom dengan membuat anotasi. `AddressKelas` dan `PhoneNumber` kelas yang ditampilkan sebelumnya dianotasi hanya dengan anotasi. `@DynamoDbBean` Saat `DynamoDB Enhanced Client API` membangun skema tabel untuk `Person` kelas dengan cuplikan berikut, API akan menemukan penggunaan class `PhoneNumber` dan membangun pemetaan yang sesuai agar berfungsi dengan `DynamoDB`. `Address`

```
TableSchema<Person> personTableSchema = TableSchema.fromBean(Person.class);
```

Gunakan skema bersarang

Pendekatan alternatif adalah dengan menggunakan pembangun skema tabel statis untuk masing-masing kelas seperti yang ditunjukkan dalam kode berikut.

Skema tabel untuk `PhoneNumber` kelas `Address` dan abstrak dalam arti bahwa mereka tidak dapat digunakan dengan tabel `DynamoDB`. Ini karena mereka tidak memiliki definisi untuk kunci utama. Mereka digunakan, bagaimanapun, sebagai skema bersarang dalam skema tabel untuk kelas. `Person`

Setelah baris komentar 1 dan 2 dalam definisi `PERSON_TABLE_SCHEMA`, Anda melihat kode yang menggunakan skema tabel abstrak. Penggunaan `documentOf` dalam `EnhanceType.documentOf(...)` metode tidak menunjukkan bahwa metode mengembalikan `EnhancedDocument` jenis API Dokumen yang Ditingkatkan. `documentOf(...)` Metode dalam konteks ini mengembalikan objek yang tahu bagaimana memetakan argumen kelas ke dan dari `DynamoDB` atribut tabel dengan menggunakan argumen skema tabel.

Kode skema statis

```
// Abstract table schema that cannot be used to work with a DynamoDB table,
// but can be used as a nested schema.
public static final TableSchema<Address> TABLE_SCHEMA_ADDRESS =
TableSchema.builder(Address.class)
    .newItemSupplier(Address::new)
    .addAttribute(String.class, a -> a.name("street")
        .getter(Address::getStreet)
```



```

        .setter(Address::setStreet))
    .addAttribute(String.class, a -> a.name("city")
        .getter(Address::getCity)
        .setter(Address::setCity))
    .addAttribute(String.class, a -> a.name("zipcode")
        .getter(Address::getZipCode)
        .setter(Address::setZipCode))
    .addAttribute(String.class, a -> a.name("state")
        .getter(Address::getState)
        .setter(Address::setState))
    .build();

// Abstract table schema that cannot be used to work with a DynamoDB table,
// but can be used as a nested schema.
public static final TableSchema<PhoneNumber> TABLE_SCHEMA_PHONENUMBER =
TableSchema.builder(PhoneNumber.class)
    .newItemSupplier(PhoneNumber::new)
    .addAttribute(String.class, a -> a.name("type")
        .getter(PhoneNumber::getType)
        .setter(PhoneNumber::setType))
    .addAttribute(String.class, a -> a.name("number")
        .getter(PhoneNumber::getNumber)
        .setter(PhoneNumber::setNumber))
    .build();

// A static table schema that can be used with a DynamoDB table.
// The table schema contains two nested schemas that are used to perform mapping
to/from DynamoDB.
public static final TableSchema<Person> PERSON_TABLE_SCHEMA =
    TableSchema.builder(Person.class)
        .newItemSupplier(Person::new)
        .addAttribute(Integer.class, a -> a.name("id")
            .getter(Person::getId)
            .setter(Person::setId)
            .addTag(StaticAttributeTags.primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("firstName")
            .getter(Person::getFirstName)
            .setter(Person::setFirstName))
        .addAttribute(String.class, a -> a.name("lastName")
            .getter(Person::getLastName)
            .setter(Person::setLastName))
        .addAttribute(Integer.class, a -> a.name("age")
            .getter(Person::getAge)
            .setter(Person::setAge))

```

```

        .addAttribute(EnhancedType.listOf(String.class), a ->
a.name("hobbies")
            .getter(Person::getHobbies)
            .setter(Person::setHobbies))
        .addAttribute(EnhancedType.mapOf(
            EnhancedType.of(String.class),
            // 1. Use mapping functionality of the Address table
schema.
            EnhancedType.documentOf(Address.class,
TABLE_SCHEMA_ADDRESS)), a -> a.name("addresses")
            .getter(Person::getAddresses)
            .setter(Person::setAddresses))
        .addAttribute(EnhancedType.listOf(
            // 2. Use mapping functionality of the PhoneNumber table
schema.
            EnhancedType.documentOf(PhoneNumber.class,
TABLE_SCHEMA_PHONENUMBER)), a -> a.name("phoneNumbers")
            .getter(Person::getPhoneNumbers)
            .setter(Person::setPhoneNumbers))
        .build();

```

## Atribut bersarang proyek

Untuk `query()` dan `scan()` metode, Anda dapat menentukan atribut mana yang ingin Anda kembalikan dalam hasil dengan menggunakan panggilan metode seperti `addNestedAttributeToProject()` dan `attributesToProject()`. DynamoDB Enhanced Client API mengubah parameter panggilan metode Java [menjadi ekspresi proyeksi sebelum permintaan dikirim](#).

Contoh berikut mengisi `Person` tabel dengan dua item, kemudian melakukan tiga operasi pemindaian.

Pemindaian pertama mengakses semua item dalam tabel untuk membandingkan hasilnya dengan operasi pemindaian lainnya.

Pemindaian kedua menggunakan metode [addNestedAttributeToProject\(\)](#) builder untuk mengembalikan hanya nilai `street` atribut.

Operasi pemindaian ketiga menggunakan metode [attributesToProject\(\)](#) builder untuk mengembalikan data untuk atribut tingkat pertama, `hobbies`. Jenis atribut `hobbies` adalah daftar. Untuk mengakses item daftar individual, lakukan `get()` operasi pada daftar.

```
    personDynamoDbTable = getDynamoDbEnhancedClient().table("Person",
PERSON_TABLE_SCHEMA);
    PersonUtils.createPersonTable(personDynamoDbTable, getDynamoDbClient());
    // Use a utility class to add items to the Person table.
    List<Person> personList = PersonUtils.getItemsForCount(2);
    // This utility method performs a put against DynamoDB to save the instances in
the list argument.
    PersonUtils.putCollection(getDynamoDbEnhancedClient(), personList,
personDynamoDbTable);

    // The first scan logs all items in the table to compare to the results of the
subsequent scans.
    final PageIterable<Person> allItems = personDynamoDbTable.scan();
    allItems.items().forEach(p ->
        // 1. Log what is in the table.
        logger.info(p.toString()));

    // Scan for nested attributes.
    PageIterable<Person> streetScanResult = personDynamoDbTable.scan(b -> b
        // Use the 'addNestedAttributeToProject()' or
'addNestedAttributesToProject()' to access data nested in maps in DynamoDB.
        .addNestedAttributeToProject(
            NestedAttributeName.create("addresses", "work", "street")
        ));

    streetScanResult.items().forEach(p ->
        //2. Log the results of requesting nested attributes.
        logger.info(p.toString()));

    // Scan for a top-level list attribute.
    PageIterable<Person> phoneNumbersScanResult = personDynamoDbTable.scan(b -> b
        // Use the 'attributesToProject()' method to access first-level
attributes.
        .attributesToProject("hobbies"));

    phoneNumbersScanResult.items().forEach((p) -> {
        // 3. Log the results of the request for the 'hobbies' attribute.
        logger.info(p.toString());
        // To access an item in a list, first get the parent attribute, 'hobbies',
then access items in the list.
        String hobby = p.getHobbies().get(1);
        // 4. Log an item in the list.
        logger.info(hobby);
    });
```

```
});
```

```
// Logged results from comment line 1.
Person{id=2, firstName='first name 2', lastName='last name 2', age=11,
  addresses={work=Address{street='street 21', city='city 21', state='state 21',
  zipCode='33333'}, home=Address{street='street 2', city='city 2', state='state 2',
  zipCode='22222'}}, phoneNumbers=[PhoneNumber{type='home', number='222-222-2222'},
  PhoneNumber{type='work', number='333-333-3333'}], hobbies=[hobby 2, hobby 21]}
Person{id=1, firstName='first name 1', lastName='last name 1', age=11,
  addresses={work=Address{street='street 11', city='city 11', state='state 11',
  zipCode='22222'}, home=Address{street='street 1', city='city 1', state='state 1',
  zipCode='11111'}}, phoneNumbers=[PhoneNumber{type='home', number='111-111-1111'},
  PhoneNumber{type='work', number='222-222-2222'}], hobbies=[hobby 1, hobby 11]}

// Logged results from comment line 2.
Person{id=null, firstName='null', lastName='null', age=null,
  addresses={work=Address{street='street 21', city='null', state='null',
  zipCode='null'}}, phoneNumbers=null, hobbies=null}
Person{id=null, firstName='null', lastName='null', age=null,
  addresses={work=Address{street='street 11', city='null', state='null',
  zipCode='null'}}, phoneNumbers=null, hobbies=null}

// Logged results from comment lines 3 and 4.
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
  phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
hobby 21
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
  phoneNumbers=null, hobbies=[hobby 1, hobby 11]}
hobby 11
```

### Note

Jika `attributesToProject()` metode mengikuti metode pembangun lain yang menambahkan atribut yang ingin Anda proyeksikan, daftar nama atribut yang diberikan ke `attributesToProject()` menggantikan semua nama atribut lainnya.

Pemindaian yang dilakukan dengan `ScanEnhancedRequest` instance dalam cuplikan berikut hanya mengembalikan data hobi.

```
ScanEnhancedRequest lastOverwrites = ScanEnhancedRequest.builder()
    .addNestedAttributeToProject(
        NestedAttributeName.create("addresses", "work", "street"))
```

```

        .addAttributeToProject("firstName")
        // If the 'attributesToProject()' method follows other builder methods
        that add attributes for projection,
        // its list of attributes replace all previous attributes.
        .attributesToProject("hobbies")
        .build();
PageIterable<Person> hobbiesOnlyResult =
    personDynamoDbTable.scan(lastOverwrites);
hobbiesOnlyResult.items().forEach(p ->
    logger.info(p.toString()));

// Logged results.
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
    phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
    phoneNumbers=null, hobbies=[hobby 1, hobby 11]}

```

Cuplikan kode berikut menggunakan `attributesToProject()` metode terlebih dahulu. Urutan ini mempertahankan semua atribut lain yang diminta.

```

ScanEnhancedRequest attributesPreserved = ScanEnhancedRequest.builder()
    // Use 'attributesToProject()' first so that the method call does not
    replace all other attributes
    // that you want to project.
    .attributesToProject("firstName")
    .addNestedAttributeToProject(
        NestedAttributeName.create("addresses", "work", "street"))
    .addAttributeToProject("hobbies")
    .build();
PageIterable<Person> allAttributesResult =
    personDynamoDbTable.scan(attributesPreserved);
allAttributesResult.items().forEach(p ->
    logger.info(p.toString()));

// Logged results.
Person{id=null, firstName='first name 2', lastName='null', age=null,
    addresses={work=Address{street='street 21', city='null', state='null',
    zipCode='null'}}, phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
Person{id=null, firstName='first name 1', lastName='null', age=null,
    addresses={work=Address{street='street 11', city='null', state='null',
    zipCode='null'}}, phoneNumbers=null, hobbies=[hobby 1, hobby 11]}

```

## Pertahankan benda kosong dengan `@DynamoDbPreserveEmptyObject`

Jika Anda menyimpan kacang ke Amazon DynamoDB dengan objek kosong dan Anda ingin SDK membuat ulang objek kosong saat pengambilan, beri anotasi pengambil kacang bagian dalam dengan `@DynamoDbPreserveEmptyObject`

Untuk mengilustrasikan cara kerja anotasi, contoh kode menggunakan dua kacang berikut.

### Contoh kacang

Kelas data berikut berisi dua InnerBean bidang. Metode `getInnerBeanWithoutAnno()`, tidak dijelaskan dengan `@DynamoDbPreserveEmptyObject` `getInnerBeanWithAnno()` Metode ini dijelaskan.

```
@DynamoDbBean
public class MyBean {

    private String id;
    private String name;
    private InnerBean innerBeanWithoutAnno;
    private InnerBean innerBeanWithAnno;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public InnerBean getInnerBeanWithoutAnno() { return innerBeanWithoutAnno; }
    public void setInnerBeanWithoutAnno(InnerBean innerBeanWithoutAnno)
{ this.innerBeanWithoutAnno = innerBeanWithoutAnno; }

    @DynamoDbPreserveEmptyObject
    public InnerBean getInnerBeanWithAnno() { return innerBeanWithAnno; }
    public void setInnerBeanWithAnno(InnerBean innerBeanWithAnno)
{ this.innerBeanWithAnno = innerBeanWithAnno; }

    @Override
    public String toString() {
        return new StringJoiner(", ", MyBean.class.getSimpleName() + "[", "]")
            .add("innerBeanWithoutAnno=" + innerBeanWithoutAnno)
            .add("innerBeanWithAnno=" + innerBeanWithAnno)
            .toString();
    }
}
```

```

        .add("id='" + id + "'")
        .add("name='" + name + "'")
        .toString();
    }
}

```

Contoh dari InnerBean kelas berikut adalah bidang MyBean dan diinisialisasi sebagai objek kosong dalam kode contoh.

```

@DynamoDbBean
public class InnerBean {

    private String innerBeanField;

    public String getInnerBeanField() {
        return innerBeanField;
    }

    public void setInnerBeanField(String innerBeanField) {
        this.innerBeanField = innerBeanField;
    }

    @Override
    public String toString() {
        return "InnerBean{" +
            "innerBeanField='" + innerBeanField + '\'' +
            '}';
    }
}

```

Contoh kode berikut menyimpan MyBean objek dengan kacang dalam yang diinisialisasi ke DynamoDB dan kemudian mengambil item tersebut. Output yang dicatat menunjukkan bahwa tidak innerBeanWithoutAnno diinisialisasi, tetapi innerBeanWithAnno telah dibuat.

```

public MyBean preserveEmptyObjectAnnoUsingGetItemExample(DynamoDbTable<MyBean>
myBeanTable) {
    // Save an item to DynamoDB.
    MyBean bean = new MyBean();
    bean.setId("1");
    bean.setInnerBeanWithoutAnno(new InnerBean()); // Instantiate the inner bean.
    bean.setInnerBeanWithAnno(new InnerBean()); // Instantiate the inner bean.
    myBeanTable.putItem(bean);
}

```

```

    GetItemEnhancedRequest request = GetItemEnhancedRequest.builder()
        .key(Key.builder().partitionValue("1").build())
        .build();
    MyBean myBean = myBeanTable.getItem(request);

    logger.info(myBean.toString());
    // Output 'MyBean[innerBeanWithoutAnno=null,
    innerBeanWithAnno=InnerBean{innerBeanField='null'}, id='1', name='null']'.

    return myBean;
}

```

## Skema statis alternatif

Anda dapat menggunakan `StaticTableSchema` versi skema tabel berikut sebagai pengganti anotasi pada kacang.

```

public static TableSchema<MyBean> buildStaticSchemas() {

    StaticTableSchema<InnerBean> innerBeanStaticTableSchema =
        StaticTableSchema.builder(InnerBean.class)
            .newItemSupplier(InnerBean::new)
            .addAttribute(String.class, a -> a.name("innerBeanField")
                .getter(InnerBean::getInnerBeanField)
                .setter(InnerBean::setInnerBeanField))
            .build();

    return StaticTableSchema.builder(MyBean.class)
        .newItemSupplier(MyBean::new)
        .addAttribute(String.class, a -> a.name("id")
            .getter(MyBean::getId)
            .setter(MyBean::setId)
            .addTag(primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("name")
            .getter(MyBean::getName)
            .setter(MyBean::setName))
        .addAttribute(EnhancedType.documentOf(InnerBean.class,
            innerBeanStaticTableSchema),
            a -> a.name("innerBean1")
                .getter(MyBean::getInnerBeanWithoutAnno)
                .setter(MyBean::setInnerBeanWithoutAnno))
        .addAttribute(EnhancedType.documentOf(InnerBean.class,

```



```

        innerBeanStaticTableSchema,
        b -> b.preserveEmptyObject(true)),
    a -> a.name("innerBean2")
        .getter(MyBean::getInnerBeanWithAnno)
        .setter(MyBean::setInnerBeanWithAnno))
    .build();
}

```

Hindari menyimpan atribut null dari objek bersarang

Anda dapat melewati atribut null dari objek bersarang saat menyimpan objek kelas data ke DynamoDB dengan menerapkan anotasi `@DynamoDbIgnoreNulls`. Sebaliknya, atribut tingkat atas dengan nilai null tidak pernah disimpan ke database.

Untuk mengilustrasikan cara kerja anotasi, contoh kode menggunakan dua kacang berikut.

Contoh kacang

Kelas data berikut berisi dua `InnerBean` bidang. Metode `getter`, `getInnerBeanWithoutAnno()`, tidak dijelaskan. `getInnerBeanWithIgnoreNullsAnno()` Metode ini dianotasi dengan `@DynamoDbIgnoreNulls`

```

@DynamoDbBean
public class MyBean {

    private String id;
    private String name;
    private InnerBean innerBeanWithoutAnno;
    private InnerBean innerBeanWithIgnoreNullsAnno;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public InnerBean getInnerBeanWithoutAnno() { return innerBeanWithoutAnno; }
    public void setInnerBeanWithoutAnno(InnerBean innerBeanWithoutAnno)
    { this.innerBeanWithoutAnno = innerBeanWithoutAnno; }

    @DynamoDbIgnoreNulls

```

```

    public InnerBean getInnerBeanWithIgnoreNullsAnno() { return
innerBeanWithIgnoreNullsAnno; }
    public void setInnerBeanWithIgnoreNullsAnno(InnerBean innerBeanWithAnno)
{ this.innerBeanWithIgnoreNullsAnno = innerBeanWithAnno; }

    @Override
    public String toString() {
        return new StringJoiner(", ", MyBean.class.getSimpleName() + "[", "]")
            .add("innerBeanWithoutAnno=" + innerBeanWithoutAnno)
            .add("innerBeanWithIgnoreNullsAnno=" + innerBeanWithIgnoreNullsAnno)
            .add("id='" + id + "'")
            .add("name='" + name + "'")
            .toString();
    }
}
}

```

Contoh dari InnerBean kelas berikut adalah bidang MyBean dan digunakan dalam kode contoh berikut.

```

@DynamoDbBean
public class InnerBean {

    private String innerBeanFieldString;
    private Integer innerBeanFieldInteger;

    public String getInnerBeanFieldString() { return innerBeanFieldString; }
    public void setInnerBeanFieldString(String innerBeanFieldString)
{ this.innerBeanFieldString = innerBeanFieldString; }

    public Integer getInnerBeanFieldInteger() { return innerBeanFieldInteger; }
    public void setInnerBeanFieldInteger(Integer innerBeanFieldInteger)
{ this.innerBeanFieldInteger = innerBeanFieldInteger; }

    @Override
    public String toString() {
        return new StringJoiner(", ", InnerBean.class.getSimpleName() + "[", "]")
            .add("innerBeanFieldString='" + innerBeanFieldString + "'")
            .add("innerBeanFieldInteger=" + innerBeanFieldInteger)
            .toString();
    }
}
}

```

Contoh kode berikut menciptakan `InnerBean` objek dan menetapkan hanya satu dari dua atribut dengan nilai.

```
public void ignoreNullsAnnoUsingPutItemExample(DynamoDbTable<MyBean> myBeanTable) {
    // Create an InnerBean object and give only one attribute a value.
    InnerBean innerBeanOneAttributeSet = new InnerBean();
    innerBeanOneAttributeSet.setInnerBeanFieldInteger(200);

    // Create a MyBean instance and use the same InnerBean instance both for
attributes.
    MyBean bean = new MyBean();
    bean.setId("1");
    bean.setInnerBeanWithoutAnno(innerBeanOneAttributeSet);
    bean.setInnerBeanWithIgnoreNullsAnno(innerBeanOneAttributeSet);

    Map<String, AttributeValue> itemMap = myBeanTable.tableSchema().itemToMap(bean,
true);
    logger.info(itemMap.toString());
    // Log the map that is sent to the database.
    //
{innerBeanWithIgnoreNullsAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200)}),
id=AttributeValue(S=1),
innerBeanWithoutAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200),
innerBeanFieldString=AttributeValue(NUL=true)}})

    // Save the MyBean object to the table.
    myBeanTable.putItem(bean);
}
```

Untuk memvisualisasikan data tingkat rendah yang dikirim ke DynamoDB, kode mencatat peta atribut sebelum menyimpan objek. `MyBean`

Output yang dicatat menunjukkan bahwa `innerBeanWithIgnoreNullsAnno` output satu atribut,

```
innerBeanWithIgnoreNullsAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200)})
```

`innerBeanWithoutAnno` menghasilkan dua atribut. Satu atribut memiliki nilai 200 dan yang lainnya adalah atribut bernilai nol.

```
innerBeanWithoutAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200),
innerBeanFieldString=AttributeValue(NUL=true)})
```

## Representasi JSON dari peta atribut

Representasi JSON berikut membuatnya lebih mudah untuk melihat data yang disimpan ke DynamoDB.

```
{
  "id": {
    "S": "1"
  },
  "innerBeanWithIgnoreNullsAnno": {
    "M": {
      "innerBeanFieldInteger": {
        "N": "200"
      }
    }
  },
  "innerBeanWithoutAnno": {
    "M": {
      "innerBeanFieldInteger": {
        "N": "200"
      },
      "innerBeanFieldString": {
        "NULL": true
      }
    }
  }
}
```

## Skema statis alternatif

Anda dapat menggunakan `StaticTableSchema` versi berikut dari skema tabel di tempat anotasi kelas data.

```
public static TableSchema<MyBean> buildStaticSchemas() {

    StaticTableSchema<InnerBean> innerBeanStaticTableSchema =
        StaticTableSchema.builder(InnerBean.class)
            .newItemSupplier(InnerBean::new)
            .addAttribute(String.class, a -> a.name("innerBeanFieldString")
                .getter(InnerBean::getInnerBeanFieldString)
                .setter(InnerBean::setInnerBeanFieldString))
            .addAttribute(Integer.class, a -> a.name("innerBeanFieldInteger")
                .getter(InnerBean::getInnerBeanFieldInteger))
}
```

```
        .setter(InnerBean::setInnerBeanFieldInteger))
        .build();

return StaticTableSchema.builder(MyBean.class)
    .newItemSupplier(MyBean::new)
    .addAttribute(String.class, a -> a.name("id")
        .getter(MyBean::getId)
        .setter(MyBean::setId)
        .addTag(primaryPartitionKey()))
    .addAttribute(String.class, a -> a.name("name")
        .getter(MyBean::getName)
        .setter(MyBean::setName))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema),
        a -> a.name("innerBeanWithoutAnno")
            .getter(MyBean::getInnerBeanWithoutAnno)
            .setter(MyBean::setInnerBeanWithoutAnno))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema,
        b -> b.ignoreNulls(true)),
        a -> a.name("innerBeanWithIgnoreNullsAnno")
            .getter(MyBean::getInnerBeanWithIgnoreNullsAnno)
            .setter(MyBean::setInnerBeanWithIgnoreNullsAnno))
    .build();
}
```

## Bekerja dengan dokumen JSON dengan Enhanced Document API untuk DynamoDB

[Enhanced Document API](#) for AWS SDK for Java 2.x dirancang untuk bekerja dengan data berorientasi dokumen yang tidak memiliki skema tetap. Namun, ini juga memungkinkan Anda menggunakan kelas khusus untuk memetakan atribut individual.

Enhanced Document API adalah penerus [Document API](#) dari AWS SDK for Java v1.x.

### Daftar Isi

- [Mulai menggunakan Enhanced Document API](#)
  - [Buat DocumentTableSchema dan a DynamoDbTable](#)
- [Membangun dokumen yang disempurnakan](#)
  - [Membangun dari string JSON](#)
  - [Membangun dari elemen individu](#)
- [Lakukan operasi CRUD](#)

- [Akses atribut dokumen yang disempurnakan sebagai objek khusus](#)
- [Gunakan EnhancedDocument tanpa DynamoDB](#)

Mulai menggunakan Enhanced Document API

Enhanced Document API memerlukan [dependensi](#) yang sama yang diperlukan untuk DynamoDB Enhanced Client API. Ini juga membutuhkan [DynamoDbEnhancedClient](#) contoh seperti yang ditunjukkan di awal topik ini.

Karena API Dokumen yang Ditingkatkan dirilis dengan versi 2.20.3 AWS SDK for Java 2.x, Anda memerlukan versi itu atau lebih tinggi.

### Buat **DocumentTableSchema** dan a **DynamoDbTable**

Untuk menjalankan perintah terhadap tabel DynamoDB menggunakan Enhanced Document API, kaitkan tabel dengan objek resource < > sisi [DynamoDbTable](#) klien [EnhancedDocument](#).

`table()` Metode klien yang disempurnakan membuat `DynamoDbTable<EnhancedDocument>` instance dan membutuhkan parameter untuk nama tabel DynamoDB dan a.

`DocumentTableSchema`

Pembangun untuk a [DocumentTableSchema](#) memerlukan kunci indeks utama dan satu atau lebih penyedia konverter atribut. `AttributeConverterProvider.defaultProvider()` Metode ini menyediakan konverter untuk [tipe default](#). Itu harus ditentukan bahkan jika Anda menyediakan penyedia konverter atribut khusus. Anda dapat menambahkan kunci indeks sekunder opsional ke pembangun.

Cuplikan kode berikut menunjukkan kode yang menghasilkan representasi sisi klien dari tabel DynamoDB yang menyimpan objek tanpa skema. `person` EnhancedDocument

```
DynamoDbTable<EnhancedDocument> documentDynamoDbTable =
    enhancedClient.table("person",
        TableSchema.documentSchemaBuilder()
            // Specify the primary key attributes.

    .addIndexPartitionKey(TableMetadata.primaryIndexName(),"id", AttributeValueType.S)
        .addIndexSortKey(TableMetadata.primaryIndexName(),
"lastName", AttributeValueType.S)
            // Specify attribute converter providers. Minimally add the
default one.
```

```
.attributeConverterProviders(AttributeConverterProvider.defaultProvider())
    .build());

// Call documentTable.createTable() if "person" does not exist in DynamoDB.
// createTable() should be called only one time.
```

Berikut ini menunjukkan representasi JSON dari person objek yang digunakan di seluruh bagian ini.

### Objek JSON **person**

```
{
  "id": 1,
  "firstName": "Richard",
  "lastName": "Roe",
  "age": 25,
  "addresses":
  {
    "home": {
      "zipCode": "00000",
      "city": "Any Town",
      "state": "FL",
      "street": "123 Any Street"
    },
    "work": {
      "zipCode": "00001",
      "city": "Anywhere",
      "state": "FL",
      "street": "100 Main Street"
    }
  },
  "hobbies": [
    "Hobby 1",
    "Hobby 2"
  ],
  "phoneNumbers": [
    {
      "type": "Home",
      "number": "555-0100"
    },
    {
      "type": "Work",
      "number": "555-0119"
    }
  ]
}
```

```
]
}
```

## Membangun dokumen yang disempurnakan

An [EnhancedDocument](#) merupakan objek tipe dokumen yang memiliki struktur kompleks dengan atribut bersarang. Sebuah EnhancedDocument membutuhkan atribut tingkat atas yang cocok dengan atribut kunci utama yang ditentukan untuk atribut. DocumentTableSchema Konten yang tersisa bersifat arbitrer dan dapat terdiri dari atribut tingkat atas dan juga atribut yang sangat bersarang.

Anda membuat EnhancedDocument instance dengan menggunakan builder yang menyediakan beberapa cara untuk menambahkan elemen.

## Membangun dari string JSON

Dengan string JSON, Anda dapat membangun panggilan EnhancedDocument dalam satu metode. Cuplikan berikut membuat EnhancedDocument dari string JSON dikembalikan oleh metode helper. `jsonPerson()` `jsonPerson()` Metode mengembalikan versi string JSON dari [objek orang](#) yang ditunjukkan sebelumnya.

```
EnhancedDocument document =
    EnhancedDocument.builder()
        .json( jsonPerson() )
        .build();
```

## Membangun dari elemen individu

Atau, Anda dapat membangun sebuah EnhancedDocument instance dari komponen individual menggunakan metode type-safe dari builder.

Contoh berikut membangun dokumen yang person disempurnakan mirip dengan dokumen yang disempurnakan yang dibangun dari string JSON pada contoh sebelumnya.

```
/* Define the shape of an address map whose JSON representation looks like the
following.
Use 'addressMapEnhancedType' in the following EnhancedDocument.builder() to
simplify the code.
"home": {
    "zipCode": "00000",
    "city": "Any Town",
```



```

        "state": "FL",
        "street": "123 Any Street"
    }*/
    EnhancedType<Map<String, String>> addressMapEnhancedType =
        EnhancedType.mapOf(EnhancedType.of(String.class),
        EnhancedType.of(String.class));

    // Use the builder's typesafe methods to add elements to the enhanced
    document.
    EnhancedDocument personDocument = EnhancedDocument.builder()
        .putNumber("id", 50)
        .putString("firstName", "Shirley")
        .putString("lastName", "Rodriguez")
        .putNumber("age", 53)
        .putNull("nullAttribute")
        .putJson("phoneNumbers", phoneNumbersJSONString())
        /* Add the map of addresses whose JSON representation looks like the
    following.
        {
            "home": {
                "zipCode": "00000",
                "city": "Any Town",
                "state": "FL",
                "street": "123 Any Street"
            }
        } */
        .putMap("addresses", getAddresses(), EnhancedType.of(String.class),
        addressMapEnhancedType)
        .putList("hobbies", List.of("Theater", "Golf"),
        EnhancedType.of(String.class))
        .build();

```

## Metode pembantu

```

private static String phoneNumbersJSONString() {
    return "[" +
        "{" +
        "  \"type\": \"Home\", " +
        "  \"number\": \"555-0140\"" +
        "}," +
        "{" +
        "  \"type\": \"Work\", " +

```

```

        "        \"number\": \"555-0155\"" +
        "    }" +
        " ]";
    }

    private static Map<String, Map<String, String>> getAddresses() {
        return Map.of(
            "home", Map.of(
                "zipCode", "00002",
                "city", "Any Town",
                "state", "ME",
                "street", "123 Any Street"));
    }

```

## Lakukan operasi CRUD

Setelah Anda mendefinisikan sebuah `EnhancedDocument` instance, Anda dapat menyimpannya ke tabel DynamoDB. Cuplikan kode berikut menggunakan [personDocument](#) yang dibuat dari elemen individual.

```
documentDynamoDbTable.putItem(personDocument);
```

Setelah Anda membaca instance dokumen yang disempurnakan dari DynamoDB, Anda dapat mengekstrak nilai atribut individual menggunakan getter seperti yang ditunjukkan dalam cuplikan kode berikut yang mengakses data yang disimpan dari `personDocument` Atau, Anda dapat mengekstrak konten lengkap ke string JSON seperti yang ditunjukkan pada bagian terakhir dari kode contoh.

```

// Read the item.
EnhancedDocument personDocFromDb =
documentDynamoDbTable.getItem(Key.builder().partitionValue(50).build());

// Access top-level attributes.
logger.info("Name: {} {}", personDocFromDb.getString("firstName"),
personDocFromDb.getString("lastName"));
// Name: Shirley Rodriguez

// Typesafe access of a deeply nested attribute. The addressMapEnhancedType
shown previously defines the shape of an addresses map.

```

```

    Map<String, Map<String, String>> addresses =
personDocFromDb.getMap("addresses", EnhancedType.of(String.class),
addressMapEnhancedType);
    addresses.keySet().forEach(k -> logger.info(addresses.get(k).toString()));
    // {zipCode=00002, city=Any Town, street=123 Any Street, state=ME}

    // Alternatively, work with AttributeValue types checking along the way for
deeply nested attributes.
    Map<String, AttributeValue> addressesMap =
personDocFromDb.getMapOfUnknownType("addresses");
    addressesMap.keySet().forEach((String k) -> {
        logger.info("Looking at data for [{}] address", k);
        // Looking at data for [home] address
        AttributeValue value = addressesMap.get(k);
        AttributeValue cityValue = value.m().get("city");
        if (cityValue != null) {
            logger.info(cityValue.s());
            // Any Town
        }
    });

    List<AttributeValue> phoneNumbers =
personDocFromDb.getListOfUnknownType("phoneNumbers");
    phoneNumbers.forEach((AttributeValue av) -> {
        if (av.hasM()) {
            AttributeValue type = av.m().get("type");
            if (type.s() != null) {
                logger.info("Type of phone: {}", type.s());
                // Type of phone: Home
                // Type of phone: Work
            }
        }
    });

    String jsonPerson = personDocFromDb.toJson();
    logger.info(jsonPerson);
    // {"firstName":"Shirley","lastName":"Rodriguez","addresses":
{"home":{"zipCode":"00002","city":"Any Town","street":"123 Any
Street","state":"ME"}}, "hobbies":["Theater","Golf"],
    //      "id":50,"nullAttribute":null,"age":53,"phoneNumbers":
[{"number":"555-0140","type":"Home"}, {"number":"555-0155","type":"Work"}]}

```

EnhancedDocumentInstance dapat digunakan dengan metode apa pun dari [DynamoDbTable](#) atau [DynamoDbEnhancedClient](#) di tempat kelas data yang dipetakan.

Akses atribut dokumen yang disempurnakan sebagai objek khusus

Selain menyediakan API untuk membaca dan menulis atribut dengan struktur tanpa skema, API Dokumen yang Ditingkatkan memungkinkan Anda mengonversi atribut ke dan dari instance kelas khusus.

Enhanced Document API menggunakan `AttributeConverterProvider`s dan `AttributeConverter`s yang ditampilkan di bagian [konversi atribut kontrol](#) sebagai bagian dari DynamoDB Enhanced Client API.

Dalam contoh berikut, kita menggunakan a `CustomAttributeConverterProvider` dengan `AddressConverter` kelas bersarang untuk mengkonversi `Address` objek.

Contoh ini menunjukkan bahwa Anda dapat mencampur data dari kelas dan juga data dari struktur yang dibangun sesuai kebutuhan. Contoh ini juga menunjukkan bahwa kelas kustom dapat digunakan pada setiap tingkat struktur bersarang. `AddressObjek` dalam contoh ini adalah nilai yang digunakan dalam peta.

```
public static void attributeToAddressClassMappingExample(DynamoDbEnhancedClient
enhancedClient, DynamoDbClient standardClient) {
    String tableName = "customer";

    // Define the DynamoDbTable for an enhanced document.
    // The schema builder provides methods for attribute converter providers and
keys.
    DynamoDbTable<EnhancedDocument> documentDynamoDbTable =
enhancedClient.table(tableName,
        DocumentTableSchema.builder()
            // Add the CustomAttributeConverterProvider along with the
default when you build the table schema.
            .attributeConverterProviders(
                List.of(
                    new CustomAttributeConverterProvider(),
                    AttributeConverterProvider.defaultProvider()))
            .addIndexPartitionKey(TableMetadata.primaryIndexName(), "id",
AttributeValueType.N)
            .addIndexSortKey(TableMetadata.primaryIndexName(), "lastName",
AttributeValueType.S)
            .build());
    // Create the DynamoDB table if needed.
```

```
documentDynamoDbTable.createTable();
waitForTableCreation(tableName, standardClient);

// The getAddressesForCustomMappingExample() helper method that provides
'addresses' shows the use of a custom Address class
// rather than using a Map<String, Map<String, String> to hold the address
data.
Map<String, Address> addresses = getAddressesForCustomMappingExample();

// Build an EnhancedDocument instance to save an item with a mix of structures
defined as needed and static classes.
EnhancedDocument personDocument = EnhancedDocument.builder()
    .putNumber("id", 50)
    .putString("firstName", "Shirley")
    .putString("lastName", "Rodriguez")
    .putNumber("age", 53)
    .putNull("nullAttribute")
    .putJson("phoneNumbers", phoneNumbersJSONString())
    // Note the use of 'EnhancedType.of(Address.class)' instead of the more
generic
    // 'EnhancedType.mapOf(EnhancedType.of(String.class),
EnhancedType.of(String.class))' that was used in a previous example.
    .putMap("addresses", addresses, EnhancedType.of(String.class),
EnhancedType.of(Address.class))
    .putList("hobbies", List.of("Hobby 1", "Hobby 2"),
EnhancedType.of(String.class))
    .build();
// Save the item to DynamoDB.
documentDynamoDbTable.putItem(personDocument);

// Retrieve the item just saved.
EnhancedDocument srPerson =
documentDynamoDbTable.getItem(Key.builder().partitionValue(50).sortValue("Rodriguez").build());

// Access the addresses attribute.
Map<String, Address> srAddresses = srPerson.get("addresses",
    EnhancedType.mapOf(EnhancedType.of(String.class),
EnhancedType.of(Address.class)));

srAddresses.keySet().forEach(k -> logger.info(addresses.get(k).toString()));

documentDynamoDbTable.deleteTable();
```

```
// The content logged to the console shows that the saved maps were converted to
Address instances.
Address{street='123 Main Street', city='Any Town', state='NC', zipCode='00000'}
Address{street='100 Any Street', city='Any Town', state='NC', zipCode='00000'}
```

## CustomAttributeConverterProviderkode

```
public class CustomAttributeConverterProvider implements AttributeConverterProvider {

    private final Map<EnhancedType<?>, AttributeConverter<?>> converterCache =
ImmutableMap.of(
        // 1. Add AddressConverter to the internal cache.
        EnhancedType.of(Address.class), new AddressConverter());

    public static CustomAttributeConverterProvider create() {
        return new CustomAttributeConverterProvider();
    }

    // 2. The enhanced client queries the provider for attribute converters if it
    // encounters a type that it does not know how to convert.
    @SuppressWarnings("unchecked")
    @Override
    public <T> AttributeConverter<T> converterFor(EnhancedType<T> enhancedType) {
        return (AttributeConverter<T>) converterCache.get(enhancedType);
    }

    // 3. Custom attribute converter
    private class AddressConverter implements AttributeConverter<Address> {
        // 4. Transform an Address object into a DynamoDB map.
        @Override
        public AttributeValue transformFrom(Address address) {

            Map<String, AttributeValue> attributeValueMap = Map.of(
                "street", AttributeValue.fromS(address.getStreet()),
                "city", AttributeValue.fromS(address.getCity()),
                "state", AttributeValue.fromS(address.getState()),
                "zipCode", AttributeValue.fromS(address.getZipCode()));

            return AttributeValue.fromM(attributeValueMap);
        }

        // 5. Transform the DynamoDB map attribute to an Address object.
        @Override
```

```
public Address transformTo(AttributeValue attributeValue) {
    Map<String, AttributeValue> m = attributeValue.m();
    Address address = new Address();
    address.setStreet(m.get("street").s());
    address.setCity(m.get("city").s());
    address.setState(m.get("state").s());
    address.setZipCode(m.get("zipCode").s());

    return address;
}

@Override
public EnhancedType<Address> type() {
    return EnhancedType.of(Address.class);
}

@Override
public AttributeValueType attributeValueType() {
    return AttributeValueType.M;
}
}
```

## Address kelas

```
public class Address {
    private String street;
    private String city;
    private String state;
    private String zipCode;

    public Address() {
    }

    public String getStreet() {
        return this.street;
    }

    public String getCity() {
        return this.city;
    }

    public String getState() {
```

```
        return this.state;
    }

    public String getZipCode() {
        return this.zipCode;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public void setState(String state) {
        this.state = state;
    }

    public void setZipCode(String zipCode) {
        this.zipCode = zipCode;
    }
}
```

Metode pembantu yang menyediakan alamat

Metode pembantu berikut menyediakan peta yang menggunakan Address instance kustom untuk nilai daripada Map<String, String> contoh generik untuk nilai.

```
private static Map<String, Address> getAddressesForCustomMappingExample() {
    Address homeAddress = new Address();
    homeAddress.setStreet("100 Any Street");
    homeAddress.setCity("Any Town");
    homeAddress.setState("NC");
    homeAddress.setZipCode("00000");

    Address workAddress = new Address();
    workAddress.setStreet("123 Main Street");
    workAddress.setCity("Any Town");
    workAddress.setState("NC");
    workAddress.setZipCode("00000");

    return Map.of("home", homeAddress,
```



```
        "work", workAddress);
    }
```

## Gunakan **EnhancedDocument** tanpa DynamoDB

Meskipun Anda biasanya menggunakan instance `EnhancedDocument` untuk membaca dan menulis item DynamoDB tipe dokumen, itu juga dapat digunakan secara independen dari DynamoDB.

Anda dapat menggunakan kemampuan mereka `EnhancedDocuments` untuk mengkonversi antara string JSON atau objek kustom ke peta tingkat rendah `AttributeValues` seperti yang ditunjukkan dalam contoh berikut.

```
public static void conversionWithoutDynamoDbExample() {
    Address address = new Address();
    address.setCity("my city");
    address.setState("my state");
    address.setStreet("my street");
    address.setZipCode("00000");

    // Build an EnhancedDocument instance for its conversion functionality alone.
    EnhancedDocument addressEnhancedDoc = EnhancedDocument.builder()
        // Important: You must specify attribute converter providers when you
        // build an EnhancedDocument instance not used with a DynamoDB table.
        .attributeConverterProviders(new CustomAttributeConverterProvider(),
        DefaultAttributeConverterProvider.create())
        .put("addressDoc", address, Address.class)
        .build();

    // Convert address to a low-level item representation.
    final Map<String, AttributeValue> addressAsAttributeMap =
addressEnhancedDoc.getMapOfUnknownType("addressDoc");
    logger.info("addressAsAttributeMap: {}", addressAsAttributeMap.toString());

    // Convert address to a JSON string.
    String addressAsJsonString = addressEnhancedDoc.toJson("addressDoc");
    logger.info("addressAsJsonString: {}", addressAsJsonString);
    // Convert addressEnhancedDoc back to an Address instance.
    Address addressConverted = addressEnhancedDoc.get("addressDoc",
Address.class);
    logger.info("addressConverted: {}", addressConverted.toString());
}

/* Console output:
```

```

        addressAsAttributeMap: {zipCode=AttributeValue(S=00000),
state=AttributeValue(S=my state), street=AttributeValue(S=my street),
city=AttributeValue(S=my city)}
        addressAsJsonString: {"zipCode":"00000","state":"my state","street":"my
street","city":"my city"}
        addressConverted: Address{street='my street', city='my city', state='my
state', zipCode='00000'}
*/

```

### Note

Bila Anda menggunakan dokumen yang disempurnakan independen dari tabel DynamoDB, pastikan Anda secara eksplisit menetapkan penyedia konverter atribut pada pembangun. Sebaliknya, skema tabel dokumen memasok penyedia konverter ketika dokumen yang disempurnakan digunakan dengan tabel DynamoDB.

## Gunakan ekstensi

DynamoDB Enhanced Client API mendukung ekstensi plugin yang menyediakan fungsionalitas di luar operasi pemetaan. Ekstensi memiliki dua metode kait, `beforeWrite()` dan `afterRead()`. `beforeWrite()` memodifikasi operasi tulis sebelum itu terjadi, dan `afterRead()` metode memodifikasi hasil operasi baca setelah itu terjadi. Karena beberapa operasi (seperti pembaruan item) melakukan penulisan dan kemudian membaca, kedua metode hook dipanggil.

Ekstensi dimuat dalam urutan yang ditentukan dalam pembuat klien yang disempurnakan. Urutan pemuatan dapat menjadi penting karena satu ekstensi dapat bertindak berdasarkan nilai yang telah diubah oleh ekstensi sebelumnya.

API klien yang disempurnakan dilengkapi dengan satu set ekstensi plugin yang terletak di [extensions](#) paket. Secara default, klien yang disempurnakan memuat [VersionedRecordExtension](#) file dan file [AtomicCounterExtension](#). Anda dapat mengganti perilaku default dengan pembuat klien peningkatan dan memuat ekstensi apa pun. Anda juga dapat menentukan `none` jika Anda tidak ingin ekstensi default.

Jika Anda memuat ekstensi Anda sendiri, klien yang disempurnakan tidak memuat ekstensi default apa pun. Jika Anda menginginkan perilaku yang disediakan oleh salah satu ekstensi default, Anda perlu menambahkannya secara eksplisit ke daftar ekstensi.

Dalam contoh berikut, ekstensi kustom bernama `verifyChecksumExtension` dimuat setelah `VersionedRecordExtension`, yang biasanya dimuat secara default dengan sendirinya. `AtomicCounterExtensionTidak` dimuat dalam contoh ini.

```
DynamoDbEnhancedClientExtension versionedRecordExtension =
    VersionedRecordExtension.builder().build();

DynamoDbEnhancedClient enhancedClient =
    DynamoDbEnhancedClient.builder()
        .dynamoDbClient(dynamoDbClient)
        .extensions(versionedRecordExtension,
verifyChecksumExtension)
        .build();
```

## VersionedRecordExtension

`VersionedRecordExtension` ini dimuat secara default dan akan menambah dan melacak nomor versi item sebagai item ditulis ke database. Sebuah kondisi akan ditambahkan ke setiap penulisan yang menyebabkan penulisan gagal jika nomor versi item tetap yang sebenarnya tidak cocok dengan nilai yang terakhir dibaca aplikasi. Perilaku ini secara efektif memberikan penguncian optimis untuk pembaruan item. Jika proses lain memperbarui item antara waktu proses pertama membaca item dan menulis pembaruan untuk itu, penulisan akan gagal.

Untuk menentukan atribut mana yang akan digunakan untuk melacak nomor versi item, beri tag atribut numerik dalam skema tabel.

Cuplikan berikut menentukan bahwa `version` atribut harus menyimpan nomor versi item.

```
@DynamoDbVersionAttribute
public Integer getVersion() {...};
public void setVersion(Integer version) {...};
```

Pendekatan skema tabel statis setara ditunjukkan dalam cuplikan berikut.

```
.addAttribute(Integer.class, a -> a.name("version")
    .getter(Customer::getVersion)
    .setter(Customer::setVersion)
    // Apply the 'version' tag to the attribute.

.tags(VersionedRecordExtension.AttributeTags.versionAttribute())
```

## AtomicCounterExtension

`AtomicCounterExtension` ini dimuat secara default dan menambah atribut numerik yang ditandai setiap kali catatan ditulis ke database. Nilai mulai dan kenaikan dapat ditentukan. Jika tidak ada nilai yang ditentukan, nilai awal diatur ke 0 dan nilai atribut meningkat sebesar 1.

Untuk menentukan atribut mana yang merupakan penghitung, beri tag atribut tipe Long dalam skema tabel.

Cuplikan berikut menunjukkan penggunaan nilai awal dan kenaikan default untuk atribut `counter`

```
@DynamoDbAtomicCounter
public Long getCounter() {...};
public void setCounter(Long counter) {...};
```

Pendekatan skema tabel statis ditunjukkan dalam cuplikan berikut. Ekstensi penghitung atom menggunakan nilai awal 10 dan menambah nilai sebesar 5 setiap kali catatan ditulis.

```
.addAttribute(Integer.class, a -> a.name("counter")
    .getter(Customer::getCounter)
    .setter(Customer::setCounter)
    // Apply the 'atomicCounter' tag to the
attribute with start and increment values.
    .tags(StaticAttributeTags.atomicCounter(10L,
5L))
```

## AutoGeneratedTimestampRecordExtension

`AutoGeneratedTimestampRecordExtension` secara otomatis memperbarui atribut tipe yang ditandai [Instant](#) dengan stempel waktu saat ini setiap kali item berhasil ditulis ke database.

Ekstensi ini tidak dimuat secara default. Oleh karena itu, Anda perlu menentukannya sebagai ekstensi khusus saat Anda membangun klien yang disempurnakan seperti yang ditunjukkan pada contoh pertama dalam topik ini.

Untuk menentukan atribut mana yang akan diperbarui dengan stempel waktu saat ini, beri tag `Instant` atribut dalam skema tabel.

`LastUpdateAtribut` adalah target perilaku ekstensi dalam cuplikan berikut. Perhatikan persyaratan bahwa atribut harus berupa `Instant` tipe.

```
@DynamoDbAutoGeneratedTimestampAttribute
```

```
public Instant getLastUpdate() {...}
public void setLastUpdate(Instant lastUpdate) {...}
```

Pendekatan skema tabel statis setara ditunjukkan dalam cuplikan berikut.

```
.addAttribute(Instant.class, a -> a.name("lastUpdate")
                .getter(Customer::getLastUpdate)
                .setter(Customer::setLastUpdate)
                // Applying the 'autoGeneratedTimestamp' tag to
the attribute.

.tags(AutoGeneratedTimestampRecordExtension.AttributeTags.autoGeneratedTimestampAttribute())
```

## Ekstensi kustom

Kelas ekstensi kustom berikut menunjukkan `beforeWrite()` metode yang menggunakan ekspresi pembaruan. Setelah baris komentar 2, kita membuat `SetAction` untuk mengatur `registrationDate` atribut jika item dalam database belum memiliki `registrationDate` atribut. Setiap kali `Customer` objek diperbarui, ekstensi memastikan bahwa `registrationDate` disetel.

```
public final class CustomExtension implements DynamoDbEnhancedClientExtension {

    // 1. In a custom extension, use an UpdateExpression to define what action to take
before
//    an item is updated.
@Override
public WriteModification beforeWrite(DynamoDbExtensionContext.BeforeWrite context)
{
    if ( context.operationContext().tableName().equals("Customer")
        && context.operationName().equals(OperationName.UPDATE_ITEM)) {
        return WriteModification.builder()
            .updateExpression(createUpdateExpression())
            .build();
    }
    return WriteModification.builder().build(); // Return an "empty"
WriteModification instance if the extension should not be applied.
// In this case, if the code is
not updating an item on the Customer table.
}

private static UpdateExpression createUpdateExpression() {
```

```

    // 2. Use a SetAction, a subclass of UpdateAction, to provide the values in the
    update.
    SetAction setAction =
        SetAction.builder()
            .path("registrationDate")
            .value("if_not_exists(registrationDate, :regValue)")
            .putExpressionValue(":regValue",
AttributeValue.fromS(Instant.now().toString()))
            .build();
    // 3. Build the UpdateExpression with one or more UpdateAction.
    return UpdateExpression.builder()
        .addAction(setAction)
        .build();
}
}

```

## Gunakan DynamoDB Enhanced Client API secara asinkron

Jika aplikasi Anda memerlukan panggilan asinkron non-pemblokiran ke DynamoDB, Anda dapat menggunakan file. [DynamoDbEnhancedAsyncClient](#) Ini mirip dengan implementasi sinkron tetapi dengan perbedaan utama berikut:

1. Ketika Anda membangun `DynamoDbEnhancedAsyncClient`, Anda harus menyediakan versi asinkron dari klien standar, `DynamoDbAsyncClient`, seperti yang ditunjukkan dalam cuplikan berikut.

```

DynamoDbEnhancedAsyncClient enhancedClient =
    DynamoDbEnhancedAsyncClient.builder()
        .dynamoDbClient(dynamoDbAsyncClient)
        .build();

```

2. Metode yang mengembalikan objek data tunggal `CompletableFuture` mengembalikan hasil bukan hanya hasilnya. Aplikasi Anda kemudian dapat melakukan pekerjaan lain tanpa harus memblokir hasilnya. Cuplikan berikut menunjukkan metode `getItem()` asinkron.

```

CompletableFuture<Customer> result = customerDynamoDbTable.getItem(customer);
// Perform other work here.
return result.join(); // Now block and wait for the result.

```

3. Metode yang mengembalikan daftar hasil paginasi mengembalikan sebuah [SdkPublisher](#) alih-alih [SdkIterable](#) yang dikembalikan sinkron `DynamoDbEnhanceClient` untuk metode yang

sama. Aplikasi Anda kemudian dapat berlangganan handler ke penerbit itu untuk menangani hasil secara asinkron tanpa harus memblokir.

```
PagePublisher<Customer> results = customerDynamoDbTable.query(r ->
    r.queryConditional(keyEqualTo(k -> k.partitionValue("Smith"))));
results.subscribe(myCustomerResultsProcessor);
// Perform other work and let the processor handle the results asynchronously.
```

Untuk contoh yang lebih lengkap tentang bekerja dengan `SdkPublisher` API, lihat [contoh di bagian](#) yang membahas `scan()` metode asinkron dari panduan ini.

## Anotasi kelas data

Tabel berikut mencantumkan anotasi yang dapat digunakan pada kelas data dan menyediakan tautan ke informasi dan contoh dalam panduan ini. Tabel diurutkan dalam urutan abjad menaik dengan nama anotasi.

Anotasi kelas data yang digunakan dalam panduan ini

Nama anotasi	Anotasi berlaku untuk*	Apa yang dilakukannya	Dimana itu ditampilkan dalam panduan ini
<code>DynamoDbAtomicCounter</code>	atribut	Menambah atribut numerik yang ditandai setiap kali catatan ditulis ke database.	<a href="#">Pengantar dan diskusi.</a>
<code>DynamoDbAttribute</code>	atribut	Mendefinisikan atau mengganti nama properti kacang yang dipetakan ke atribut tabel DynamoDB.	<ul style="list-style-type: none"> <li>• <a href="#">Diskusi awal.</a></li> <li>• <a href="#">Memulai bagian—lihat Catatan.</a></li> <li>• <a href="#">Di MovieActor kelas Dalam contoh metode Query.</a></li> </ul>
<code>DynamoDbAutoGeneratedTimestampAttribute</code>	atribut	Memperbarui atribut yang ditandai dengan stempel waktu saat ini setiap kali item	<a href="#">Pengantar dan diskusi.</a>

Nama anotasi	Anotasi berlaku untuk*	Apa yang dilakukannya	Dimana itu ditampilkan dalam panduan ini
		berhasil ditulis ke database	
DynamoDbBean	class	Menandai kelas data sebagai dapat dipetakan ke skema tabel.	Pertama gunakan pada <a href="#">kelas Pelanggan</a> di bagian Memulai. Beberapa penggunaan muncul di seluruh panduan.
DynamoDbConvertedBy	atribut	Mengaitkan kustom Attribute Converter dengan atribut beranotasi.	<a href="#">Diskusi awal dan contoh.</a>
DynamoDbFlatten	atribut	Meratakan semua atribut dari kelas data DynamoDB terpisah dan menambahkannya sebagai atribut tingkat atas ke catatan yang dibaca dan ditulis ke database.	<ul style="list-style-type: none"> <li>• <a href="#">Diskusi awal.</a></li> <li>• <a href="#">Implikasi untuk kode lainnya.</a></li> </ul>
DynamoDbIgnore	atribut	Menghasilkan atribut yang tersisa tidak dipetakan.	<ul style="list-style-type: none"> <li>• <a href="#">Diskusi awal.</a></li> <li>• <a href="#">Gunakan di ProductCatalog kelas.</a></li> </ul>



Nama anotasi	Anotasi berlaku untuk*	Apa yang dilakukannya	Dimana itu ditampilkan dalam panduan ini
DynamoDbIgnoreNulls	atribut	Mencegah menyimpan atribut null dari objek bersarang DynamoDb .	<a href="#">Diskusi dan contoh.</a>
DynamoDbImmutable	class	Menandai kelas data yang tidak dapat diubah sebagai dapat dipetakan ke skema tabel.	<ul style="list-style-type: none"> <li>• <a href="#">Pengantar anotasi.</a></li> <li>• <a href="#">Gunakan di ProductCatalog kelas.</a></li> <li>• <a href="#">Gunakan dengan Lombok.</a></li> </ul>
DynamoDbPartitionKey	atribut	Menandai atribut sebagai kunci partisi primer (kunci hash) dari DynamoDb tabel.	<ul style="list-style-type: none"> <li>• <a href="#">Penggunaan awal pada kelas Pelanggan di bagian Memulai.</a></li> <li>• <a href="#">Dengan Lombok.</a></li> </ul>
DynamoDbP reserveEmptyObject	atribut	Menentukan bahwa jika tidak ada data hadir untuk objek dipetakan ke atribut beranotasi, objek harus diinisialisasi dengan semua bidang null.	<a href="#">Diskusi dan contoh.</a>

Nama anotasi	Anotasi berlaku untuk*	Apa yang dilakukannya	Dimana itu ditampilkan dalam panduan ini
DynamoDbSecondaryPartitionKey	atribut	Menandai atribut sebagai kunci partisi untuk indeks sekunder global.	<ul style="list-style-type: none"> <li>• <a href="#">Gunakan dalam indeks sekunder dan contoh.</a></li> <li>• <a href="#">Dalam contoh metode Query.</a></li> <li>• <a href="#">Dalam contoh Lombok</a></li> <li>• <a href="#">Dengan kelas yang tidak dapat diubah.</a></li> </ul>
DynamoDbSecondarySortKey	atribut	Menandai atribut sebagai kunci pengurutan opsional untuk indeks sekunder global atau lokal.	<ul style="list-style-type: none"> <li>• <a href="#">Gunakan dalam indeks sekunder dan contoh.</a></li> <li>• <a href="#">Dalam contoh metode Query.</a></li> <li>• <a href="#">Dalam contoh Lombok.</a></li> <li>• <a href="#">Dengan kelas yang tidak dapat diubah.</a></li> </ul>
DynamoDbSortKey	atribut	Menandai atribut sebagai kunci sortir primer opsional (range key).	<ul style="list-style-type: none"> <li>• <a href="#">Memulai bagian di kelas Pelanggan.</a></li> <li>• <a href="#">Dengan kelas yang tidak dapat diubah.</a></li> <li>• <a href="#">Dalam contoh Lombok.</a></li> <li>• <a href="#">Dalam contoh metode Query.</a></li> </ul>

Nama anotasi	Anotasi berlaku untuk*	Apa yang dilakukannya	Dimana itu ditampilkan dalam panduan ini
DynamoDbUpdateBehavior	atribut	Menentukan perilaku ketika atribut ini diperbarui sebagai bagian dari operasi 'update' seperti UpdateItem.	<a href="#">Pendahuluan dan contoh.</a>
DynamoDbVersionAttribute	atribut	Menambah nomor versi item.	<a href="#">Pengantar dan diskusi.</a>

\*Anda dapat menerapkan anotasi tingkat atribut ke pengambil atau penyetal, tetapi tidak keduanya. Panduan ini menunjukkan anotasi tentang getter.

## Bekerja dengan Amazon EC2

Bagian ini memberikan contoh pemrograman [Amazon EC2](#) yang menggunakan AWS SDK for Java 2.x.

Topik

- [Kelola Amazon EC2 instance](#)
- [Zona Penggunaan Wilayah AWS dan Ketersediaan](#)
- [Bekerja dengan kelompok keamanan di Amazon EC2](#)
- [Bekerja dengan metadata instans Amazon EC2](#)

### Kelola Amazon EC2 instance

Buatlah sebuah instans

Buat Amazon EC2 instance baru dengan memanggil [runInstances](#) metode [Ec2Client](#), sediakan dengan [RunInstancesRequest](#) berisi [Amazon Machine Image \(AMI\)](#) untuk digunakan dan jenis [instance](#).

Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

## Kode

```
public static String createEC2Instance(Ec2Client ec2, String name, String amiId ) {

    RunInstancesRequest runRequest = RunInstancesRequest.builder()
        .imageId(amiId)
        .instanceType(InstanceType.T1_MICRO)
        .maxCount(1)
        .minCount(1)
        .build();

    RunInstancesResponse response = ec2.runInstances(runRequest);
    String instanceId = response.instances().get(0).instanceId();

    Tag tag = Tag.builder()
        .key("Name")
        .value(name)
        .build();

    CreateTagsRequest tagRequest = CreateTagsRequest.builder()
        .resources(instanceId)
        .tags(tag)
        .build();

    try {
        ec2.createTags(tagRequest);
        System.out.printf(
            "Successfully started EC2 Instance %s based on AMI %s",
            instanceId, amiId);

        return instanceId;
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }

    return "";
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Memulai instans

Untuk memulai sebuah Amazon EC2 instance, panggil [startInstances](#) metode Ec2Client, sediakan dengan ID [StartInstancesRequest](#) yang berisi instance untuk memulai.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
```

### Kode

```
public static void startInstance(Ec2Client ec2, String instanceId) {

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    ec2.startInstances(request);
    System.out.printf("Successfully started instance %s", instanceId);
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Menghentikan instans

Untuk menghentikan Amazon EC2 instance, panggil [stopInstances](#) metode Ec2Client, berikan ID yang [StopInstancesRequest](#) berisi instance untuk berhenti.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
```

## Kode

```
public static void stopInstance(Ec2Client ec2, String instanceId) {

    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    ec2.stopInstances(request);
    System.out.printf("Successfully stopped instance %s", instanceId);
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Menyalakan ulang instans

Untuk me-reboot sebuah Amazon EC2 instance, panggil [rebootInstances](#) metode Ec2Client, berikan ID yang [RebootInstancesRequest](#) berisi instance untuk reboot.

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.RebootInstancesRequest;
```

## Kode

```
public static void rebootEC2Instance(Ec2Client ec2, String instanceId) {

    try {
        RebootInstancesRequest request = RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();
    }
```

```
        ec2.rebootInstances(request);
        System.out.printf(
            "Successfully rebooted instance %s", instanceId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Menjelaskan instans

Untuk membuat daftar instance Anda, buat [DescribeInstancesRequest](#) dan panggil metode `Ec2Client.describeInstances`. Ini akan mengembalikan [DescribeInstancesResponse](#) objek yang dapat Anda gunakan untuk daftar Amazon EC2 instance untuk akun dan wilayah Anda.

Instans dikelompokkan berdasarkan reservasi. Setiap reservasi sesuai dengan panggilan `startInstances` yang meluncurkan instance. Untuk membuat daftar instance Anda, Anda harus terlebih dahulu memanggil `reservations` metode `DescribeInstancesResponse` kelas, dan kemudian memanggil `instances` setiap objek [Reservasi](#) yang dikembalikan.

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.Reservation;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

## Kode

```
public static void describeEC2Instances( Ec2Client ec2){

    String nextToken = null;

    try {

        do {
```

```

        DescribeInstancesRequest request =
DescribeInstancesRequest.builder().maxResults(6).nextToken(nextToken).build();
        DescribeInstancesResponse response = ec2.describeInstances(request);

        for (Reservation reservation : response.reservations()) {
            for (Instance instance : reservation.instances()) {
                System.out.println("Instance Id is " + instance.instanceId());
                System.out.println("Image id is "+ instance.imageId());
                System.out.println("Instance type is "+
instance.instanceType());
                System.out.println("Instance state name is "+
instance.state().name());
                System.out.println("monitoring information is "+
instance.monitoring().state());

            }
        }
        nextToken = response.nextToken();
    } while (nextToken != null);

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

```

Hasil paged; Anda bisa mendapatkan hasil lebih lanjut dengan meneruskan nilai yang dikembalikan dari `nextToken` metode objek hasil ke `nextToken` metode objek permintaan baru, kemudian menggunakan objek permintaan baru dalam panggilan berikutnya. `describeInstances`

Lihat [contoh lengkapnya](#) di GitHub.

## Memantau sebuah instance

Anda dapat memantau berbagai aspek Amazon EC2 instance Anda, seperti CPU dan pemanfaatan jaringan, memori yang tersedia, dan ruang disk yang tersisa. Untuk mempelajari lebih lanjut tentang pemantauan instans, lihat [Pemantauan Amazon EC2](#) di Panduan Amazon EC2 Pengguna untuk Instans Linux.

Untuk mulai memantau instance, Anda harus membuat [MonitorInstancesRequest](#) dengan ID instance untuk dipantau, dan meneruskannya ke metode [monitorInstances](#) `Ec2Client`.

Impor



```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.MonitorInstancesRequest;
import software.amazon.awssdk.services.ec2.model.UnmonitorInstancesRequest;
```

## Kode

```
public static void monitorInstance( Ec2Client ec2, String instanceId) {

    MonitorInstancesRequest request = MonitorInstancesRequest.builder()
        .instanceIds(instanceId).build();

    ec2.monitorInstances(request);
    System.out.printf(
        "Successfully enabled monitoring for instance %s",
        instanceId);
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Hentikan pemantauan instans

Untuk menghentikan pemantauan instance, buat [UnmonitorInstancesRequest](#) dengan ID instance untuk menghentikan pemantauan, dan teruskan ke metode Ec2Client. [unmonitorInstances](#)

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.MonitorInstancesRequest;
import software.amazon.awssdk.services.ec2.model.UnmonitorInstancesRequest;
```

## Kode

```
public static void unmonitorInstance(Ec2Client ec2, String instanceId) {

    UnmonitorInstancesRequest request = UnmonitorInstancesRequest.builder()
        .instanceIds(instanceId).build();

    ec2.unmonitorInstances(request);
}
```

```
System.out.printf(
    "Successfully disabled monitoring for instance %s",
    instanceId);
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Informasi lain

- [RunInstances](#) di Referensi Amazon EC2 API
- [DescribeInstances](#) di Referensi Amazon EC2 API
- [StartInstances](#) di Referensi Amazon EC2 API
- [StopInstances](#) di Referensi Amazon EC2 API
- [RebootInstances](#) di Referensi Amazon EC2 API
- [MonitorInstances](#) di Referensi Amazon EC2 API
- [UnmonitorInstances](#) di Referensi Amazon EC2 API

## Zona Penggunaan Wilayah AWS dan Ketersediaan

### Menjelaskan Wilayah

Untuk mencantumkan Wilayah yang tersedia untuk akun Anda, hubungi metode `Ec2Client.describeRegions`. Ini mengembalikan a [DescribeRegionsResponse](#). Panggil `regions` metode objek yang dikembalikan untuk mendapatkan daftar objek [Region](#) yang mewakili setiap Region.

### Impor

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

### Kode

```
try {
```

```
DescribeRegionsResponse regionsResponse = ec2.describeRegions();
regionsResponse.regions().forEach(region -> {
    System.out.printf(
        "Found Region %s with endpoint %s%n",
        region.regionName(),
        region.endpoint());
    System.out.println();
});
```

Lihat [contoh lengkapnya](#) di GitHub.

## Jelaskan zona ketersediaan

Untuk mencantumkan setiap Availability Zone yang tersedia untuk akun Anda, hubungi metode `Ec2Client.describeAvailabilityZones`. Ini mengembalikan a [DescribeAvailabilityZonesResponse](#). Panggil `availabilityZones` metodenya untuk mendapatkan daftar [AvailabilityZone](#) objek yang mewakili setiap Availability Zone.

### Impor

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

### Kode

#### Buat `Ec2Client`.

```
software.amazon.awssdk.regions.Region region =
software.amazon.awssdk.regions.Region.US_EAST_1;
Ec2Client ec2 = Ec2Client.builder()
    .region(region)
    .build();
```

Kemudian panggil `describeAvailabilityZones ()` dan ambil hasilnya.

```
DescribeAvailabilityZonesResponse zonesResponse =
ec2.describeAvailabilityZones();
```

```
zonesResponse.availabilityZones().forEach(zone -> {
    System.out.printf(
        "Found Availability Zone %s with status %s in region %s%n",
        zone.zoneName(),
        zone.state(),
        zone.regionName()
    );
    System.out.println();
});
```

Lihat [contoh lengkapnya](#) di GitHub.

## Jelaskan akun

Untuk mencantumkan informasi terkait EC2 tentang akun Anda, hubungi metode `Ec2Client.describeAccountAttributes`. Metode ini mengembalikan [DescribeAccountAttributesResponse](#) objek. Memanggil `describeAccountAttributes` metode objek ini untuk mendapatkan daftar [AccountAttribute](#) objek. Anda dapat mengulangi melalui daftar untuk mengambil objek `AccountAttribute`.

Anda bisa mendapatkan nilai atribut akun Anda dengan menjalankan `getAttributeValues` metode `AccountAttribute` objek. Metode ini mengembalikan daftar [AccountAttributeValue](#) objek. Anda dapat mengulangi melalui daftar kedua ini untuk menampilkan nilai atribut (lihat contoh kode berikut).

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

## Kode

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DescribeAccount {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        describeEC2Account(ec2);
        System.out.print("Done");
        ec2.close();
    }

    public static void describeEC2Account(Ec2Client ec2) {
        try {
            DescribeAccountAttributesResponse accountResults =
ec2.describeAccountAttributes();
            accountResults.accountAttributes().forEach(attribute -> {
                System.out.print("\n The name of the attribute is " +
attribute.attributeName());
                attribute.attributeValues().forEach(
                    myValue -> System.out.print("\n The value of the attribute is "
+ myValue.attributeValue()));
            });

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Informasi lain

- [Wilayah dan Availability Zone](#) di Panduan Amazon EC2 Pengguna untuk Instans Linux
- [DescribeRegions](#) di Referensi Amazon EC2 API
- [DescribeAvailabilityZones](#) di Referensi Amazon EC2 API

## Bekerja dengan kelompok keamanan di Amazon EC2

### Membuat grup keamanan

Untuk membuat grup keamanan, panggil `createSecurityGroup` metode `Ec2Client` dengan [CreateSecurityGroupRequest](#) yang berisi nama kunci.

#### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.IpRange;
```

#### Kode

```
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
    .groupName(groupName)
    .description(groupDesc)
    .vpcId(vpcId)
    .build();

        CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);
```

Lihat [contoh lengkapnya](#) di GitHub.

### Konfigurasi grup keamanan

Grup keamanan dapat mengontrol lalu lintas masuk (masuk) dan keluar (keluar) ke instans Anda. Amazon EC2

Untuk menambahkan aturan ingress ke grup keamanan Anda, gunakan `authorizeSecurityGroupIngress` metode `Ec2Client`, dengan memberikan nama grup keamanan dan aturan akses ([IpPermission](#)) yang ingin Anda tetapkan di dalam objek. [AuthorizeSecurityGroupIngressRequest](#) Contoh berikut menunjukkan cara menambahkan izin IP ke grup keamanan.

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.IpRange;
```

## Kode

### Pertama, buat Ec2Client

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

### Kemudian gunakan metode Ec2Client, authorizeSecurityGroupIngress

```
IpRange ipRange = IpRange.builder()
    .cidrIp("0.0.0.0/0").build();

IpPermission ipPerm = IpPermission.builder()
    .ipProtocol("tcp")
    .toPort(80)
    .fromPort(80)
    .ipRanges(ipRange)
    .build();

IpPermission ipPerm2 = IpPermission.builder()
    .ipProtocol("tcp")
    .toPort(22)
    .fromPort(22)
    .ipRanges(ipRange)
    .build();

AuthorizeSecurityGroupIngressRequest authRequest =
    AuthorizeSecurityGroupIngressRequest.builder()
```

```
        .groupName(groupName)
        .ipPermissions(ipPerm, ipPerm2)
        .build();

    AuthorizeSecurityGroupIngressResponse authResponse =
    ec2.authorizeSecurityGroupIngress(authRequest);

    System.out.printf(
        "Successfully added ingress policy to Security Group %s",
        groupName);

    return resp.groupId();

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

Untuk menambahkan aturan keluar ke grup keamanan, berikan data serupa dalam metode `Ec2Client.authorizeSecurityGroupEgressRequest` `authorizeSecurityGroupEgress`

Lihat [contoh lengkapnya](#) di GitHub.

## Jelaskan kelompok keamanan

Untuk mendeskripsikan grup keamanan Anda atau mendapatkan informasi tentang mereka, hubungi metode `Ec2Client.describeSecurityGroups` Ia mengembalikan [DescribeSecurityGroupsResponse](#) yang dapat Anda gunakan untuk mengakses daftar grup keamanan dengan memanggil `securityGroups` metodenya, yang mengembalikan daftar [SecurityGroup](#) objek.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.SecurityGroup;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```



## Kode

```
public static void describeEC2SecurityGroups(Ec2Client ec2, String groupId) {  
  
    try {  
        DescribeSecurityGroupsRequest request =  
            DescribeSecurityGroupsRequest.builder()  
                .groupIds(groupId).build();  
  
        DescribeSecurityGroupsResponse response =  
            ec2.describeSecurityGroups(request);  
  
        for(SecurityGroup group : response.securityGroups()) {  
            System.out.printf(  
                "Found Security Group with id %s, " +  
                "vpc id %s " +  
                "and description %s",  
                group.groupId(),  
                group.vpcId(),  
                group.description());  
        }  
    } catch (Ec2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Menghapus grup keamanan

Untuk menghapus grup keamanan, panggil `deleteSecurityGroup` metode `Ec2Client`, berikan [DeleteSecurityGroupRequest](#) yang berisi ID grup keamanan untuk dihapus.

### Impor

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.ec2.Ec2Client;  
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;  
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

## Kode

```
public static void deleteEC2SecGroup(Ec2Client ec2,String groupId) {

    try {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        ec2.deleteSecurityGroup(request);
        System.out.printf(
            "Successfully deleted Security Group with id %s", groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Informasi lain

- [Amazon EC2 Grup Keamanan](#) dalam Panduan Amazon EC2 Pengguna untuk Instans Linux
- [Otorisasi lalu lintas masuk untuk Instans Linux Anda](#) di Panduan Amazon EC2 Pengguna untuk Instans Linux
- [CreateSecurityGroup](#) di Referensi Amazon EC2 API
- [DescribeSecurityGroups](#) di Referensi Amazon EC2 API
- [DeleteSecurityGroup](#) di Referensi Amazon EC2 API
- [AuthorizeSecurityGroupIngress](#) di Referensi Amazon EC2 API

## Bekerja dengan metadata instans Amazon EC2

Klien Java SDK untuk Layanan Metadata Instans Amazon EC2 (klien metadata) memungkinkan aplikasi Anda mengakses metadata pada instans EC2 lokal mereka. Klien metadata bekerja dengan instance lokal [IMDSv2](#) (Instance Metadata Service v2) dan menggunakan permintaan berorientasi sesi.

Dua kelas klien tersedia di SDK. Sinkron [Ec2MetadataClient](#) adalah untuk memblokir operasi, dan [Ec2MetadataAsyncClient](#) adalah untuk kasus penggunaan asinkron dan non-pemblokiran.

## Memulai

Untuk menggunakan klien metadata, tambahkan artefak `imds` Maven ke proyek Anda. Anda juga membutuhkan kelas untuk [SdkHttpClient](#) (atau [SdkAsyncHttpClient](#) untuk varian asinkron) pada classpath.

Berikut Maven XML menunjukkan potongan ketergantungan untuk menggunakan sinkron [URLConnectionHttpClient](#) bersama dengan ketergantungan untuk klien metadata.

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

<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>imds</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>url-connection-client</artifactId>
  </dependency>
  <!-- other dependencies -->
</dependencies>
```

Cari [repositori pusat Maven](#) untuk versi terbaru daribom artefak.

Untuk menggunakan klien HTTP asynchronous, ganti potongan dependensi untuk `url-connection-client` artefak. Misalnya, cuplikan berikut membawa [NettyNioAsyncHttpClient](#) implementasi.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>netty-nio-client</artifactId>
```

```
</dependency>
```

## Menggunakan klien metadata

### Instantiate klien metadata

Anda dapat instantiate sebuah instance dari `Ec2MetadataClient` ketika hanya satu implementasi `SdkHttpClient` antarmuka hadir pada classpath. Untuk melakukannya, panggil `Ec2MetadataClient#create()` metode statis seperti yang ditunjukkan dalam potongan berikut.

```
Ec2MetadataClient client = Ec2MetadataClient.create(); //  
'Ec2MetadataAsyncClient#create' is the asynchronous version.
```

Jika aplikasi Anda memiliki beberapa implementasi `SdkHttpClient` atau `SdkHttpAsyncClient` antarmuka, Anda harus menentukan implementasi untuk metadata klien untuk digunakan seperti yang ditunjukkan pada [the section called "Klien HTTP yang dapat dikonfigurasi"](#) bagian.

#### Note

Untuk sebagian besar klien layanan, seperti Amazon S3, SDK for Java secara otomatis menambahkan implementasi `SdkHttpClient` atau `SdkHttpAsyncClient` antarmuka. Jika klien metadata Anda menggunakan implementasi yang sama, maka `Ec2MetadataClient#create()` akan bekerja. Jika Anda memerlukan implementasi yang berbeda, Anda harus menentukannya saat membuat klien metadata.

### Kirim permintaan

Untuk mengambil metadata contoh, instantiate `EC2MetadataClient` kelas dan memanggil metode dengan parameter `path` yang menentukan [kategori metadata instance](#).

Contoh berikut mencetak nilai yang terkait dengan `ami-id` kunci ke konsol.

```
Ec2MetadataClient client = Ec2MetadataClient.create();  
Ec2MetadataResponse response = client.get("/latest/meta-data/ami-id");  
System.out.println(response.asString());  
client.close(); // Closes the internal resources used by the Ec2MetadataClient class.
```

Jika jalan tidak valid, `get` metode ini akan melempar eksepsi.

Gunakan kembali instance klien yang sama untuk beberapa permintaan, tetapi hubungi `close` klien ketika tidak lagi diperlukan untuk melepaskan sumber daya. Setelah metode `close` dipanggil, instance klien tidak dapat digunakan lagi.

## Mengurai Tanggapan

EC2 contoh metadata dapat output dalam format yang berbeda. Teks biasa dan JSON adalah format yang paling umum digunakan. Klien metadata menawarkan cara untuk bekerja dengan format tersebut.

Seperti contoh berikut menunjukkan, menggunakan `asString` metode untuk mendapatkan data sebagai string Java. Anda juga dapat menggunakan `asList` metode untuk memisahkan respons teks biasa yang mengembalikan beberapa baris.

```
Ec2MetadataClient client = Ec2MetadataClient.create();
Ec2MetadataResponse response = client.get("/latest/meta-data/");
String fullResponse = response.asString();
List<String> splits = response.asList();
```

Jika responsnya ada di JSON, gunakan `Ec2MetadataResponse#asDocument` metode ini untuk mengurai respons JSON menjadi instance [Document](#) seperti yang ditunjukkan pada cuplikan kode berikut.

```
Document fullResponse = response.asDocument();
```

Pengecualian akan dilemparkan jika format metadata tidak dalam JSON. Jika respons berhasil diurai, Anda dapat menggunakan [API dokumen](#) untuk memeriksa respons secara lebih rinci. Lihat [bagan kategori metadata](#) instans untuk mempelajari kategori metadata mana yang memberikan respons berformat JSON.

## Konfigurasi klien metadata

### Percobaan ulang

Anda dapat mengkonfigurasi klien metadata dengan mekanisme coba lagi. Jika Anda melakukannya, maka klien dapat secara otomatis mencoba kembali permintaan yang gagal karena alasan yang tidak terduga. Secara default, klien mencoba ulang tiga kali pada permintaan gagal dengan waktu backoff eksponensial antara upaya.

Jika kasus penggunaan Anda memerlukan mekanisme percobaan ulang yang berbeda, Anda dapat menyesuaikan klien menggunakan `retryPolicy` metode pada pembangunnya. Misalnya, contoh berikut menunjukkan klien sinkron yang dikonfigurasi dengan penundaan tetap dua detik antara upaya dan lima upaya coba lagi.

```
BackoffStrategy fixedBackoffStrategy =
    FixedDelayBackoffStrategy.create(Duration.ofSeconds(2));
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .retryPolicy(retryPolicyBuilder ->
            retryPolicyBuilder.numRetries(5)

            .backoffStrategy(fixedBackoffStrategy))
        .build();
```

Ada beberapa [BackoffStrategies](#) yang dapat Anda gunakan dengan klien metadata.

Anda juga dapat menonaktifkan mekanisme percobaan ulang sepenuhnya, seperti yang ditunjukkan cuplikan berikut.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .retryPolicy(Ec2MetadataRetryPolicy.none())
        .build();
```

Menggunakan `Ec2MetadataRetryPolicy#none()` menonaktifkan kebijakan coba ulang default sehingga klien metadata mencoba tidak ada percobaan ulang.

## Versi IP

Secara default, klien metadata menggunakan titik akhir IPV4 di `http://169.254.169.254`. Untuk mengubah klien untuk menggunakan versi IPV6, gunakan salah satu `endpointMode` atau `endpoint` metode pembangun. Hasil pengecualian jika kedua metode dipanggil pada pembangun.

Contoh berikut ini menunjukkan kedua opsi IPV6.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .endpointMode(EndpointMode.IPV6)
        .build();
```

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .endpoint(URI.create("http://[fd00:ec2::254]"))
        .build();
```

## Fitur kunci

### Klien asinkron

Untuk menggunakan versi non-blocking klien, instantiate instance dari `Ec2MetadataAsyncClient` kelas. Kode dalam contoh berikut membuat klien asynchronous dengan pengaturan default dan menggunakan `get` metode untuk mengambil nilai untuk `ami-id` kunci.

```
Ec2MetadataAsyncClient asyncClient = Ec2MetadataAsyncClient.create();
CompletableFuture<Ec2MetadataResponse> response = asyncClient.get("/latest/meta-data/ami-id");
```

Yang `java.util.concurrent.CompletableFuture` dikembalikan oleh `get` metode selesai ketika respon kembali. Contoh berikut mencetak `ami-id` metadata ke konsol.

```
response.thenAccept(metadata -> System.out.println(metadata.asString()));
```

### Klien HTTP yang dapat dikonfigurasi

Pembangun untuk setiap klien metadata memiliki `httpClient` metode yang dapat Anda gunakan untuk memasok klien HTTP yang disesuaikan.

Contoh berikut menunjukkan kode untuk `URLConnectionHttpClient` contoh kustom.

```
SdkHttpClient httpClient =
    UrlConnectionHttpClient.builder()
        .socketTimeout(Duration.ofMinutes(5))
        .proxyConfiguration(proxy ->
            proxy.endpoint(URI.create("http://proxy.example.net:8888")))
        .build();
Ec2MetadataClient metaDataClient =
    Ec2MetadataClient.builder()
        .httpClient(httpClient)
        .build();
// Use the metaDataClient instance.
```

```
metaDataClient.close(); // Close the instance when no longer needed.
```

Contoh berikut menunjukkan kode untuk `NettyNioAsyncHttpClient` instance kustom dengan klien metadata asinkron.

```
SdkAsyncHttpClient httpAsyncClient =  
    NettyNioAsyncHttpClient.builder()  
        .connectionTimeout(Duration.ofMinutes(5))  
        .maxConcurrency(100)  
        .build();  
Ec2MetadataAsyncClient asyncMetaClient =  
    Ec2MetadataAsyncClient.builder()  
        .httpClient(httpAsyncClient)  
        .build();  
// Use the asyncMetaClient instance.  
asyncMetaClient.close(); // Close the instance when no longer needed.
```

[the section called “Klien HTTP”](#) Topik dalam panduan ini memberikan rincian tentang cara mengkonfigurasi klien HTTP yang tersedia di SDK for Java.

## Caching Token

Karena klien metadata menggunakan IMDSv2, semua permintaan dikaitkan dengan sesi. Sesi didefinisikan oleh token yang memiliki kedaluwarsa, yang dikelola klien metadata untuk Anda. Setiap permintaan metadata secara otomatis menggunakan kembali token hingga kedaluwarsa.

Secara default-nya, token berlangsung selama enam jam (21.600 detik). Kami sarankan Anda menyimpan time-to-live default-nya, kecuali kasus penggunaan spesifik Anda memerlukan konfigurasi lanjutan.

Jika diperlukan, konfigurasi durasi dengan menggunakan metode `tokenTtl` pembangun. Misalnya, kode dalam cuplikan berikut membuat klien dengan durasi sesi lima menit.

```
Ec2MetadataClient client =  
    Ec2MetadataClient.builder()  
        .tokenTtl(Duration.ofMinutes(5))  
        .build();
```

Jika Anda menghilangkan panggilan `tokenTtl` metode pada builder, durasi default 21.600 digunakan sebagai gantinya.



# Bekerja dengan IAM

Bagian ini memberikan contoh pemrograman AWS Identity and Access Management (IAM) dengan menggunakan AWS SDK for Java 2.x.

AWS Identity and Access Management(IAM) memungkinkan Anda untuk mengontrol akses ke AWS layanan dan sumber daya untuk pengguna Anda dengan aman. Dengan menggunakan IAM, Anda dapat membuat dan mengelola AWS pengguna dan grup, serta menggunakan izin untuk mengizinkan dan menolak akses mereka ke AWS sumber daya. Untuk panduan lengkap IAM, kunjungi [Panduan IAM Pengguna](#).

Contoh berikut hanya mencakup kode yang diperlukan untuk mendemonstrasikan setiap teknik. [Kode contoh lengkap tersedia di GitHub](#). Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk dibangun dan dijalankan.

Topik

- [Kelola kunci IAM akses](#)
- [Kelola IAM Pengguna](#)
- [Buat kebijakan IAM dengan AWS SDK for Java 2.x](#)
- [Bekerja dengan IAM kebijakan](#)
- [Bekerja dengan sertifikat IAM server](#)

## Kelola kunci IAM akses

Buat kunci akses

Untuk membuat kunci IAM akses, panggil `IamClient`'s `createAccessKey` metode dengan [CreateAccessKeyRequest](#) objek.

### Note

Anda harus menyetel wilayah ke `AWS_GLOBAL` agar `IamClient` panggilan berfungsi karena IAM merupakan layanan global.

Impor

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

## Kode

```
public static String createIAMAccessKey(IamClient iam,String user) {

    try {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(user).build();

        CreateAccessKeyResponse response = iam.createAccessKey(request);
        String keyId = response.accessKey().accessKeyId();
        return keyId;

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Daftar kunci akses

Untuk membuat daftar kunci akses untuk pengguna tertentu, buat [ListAccessKeysRequest](#) objek yang berisi nama pengguna untuk mencantumkan kunci, dan meneruskannya ke `IamClient`'s `listAccessKeys` metode.

### Note

Jika Anda tidak memberikan nama pengguna ke `listAccessKeys`, itu akan mencoba untuk daftar kunci akses yang terkait dengan Akun AWS yang menandatangani permintaan.

## Impor

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

## Kode

```
public static void listKeys( IamClient iam,String userName ){

    try {
        boolean done = false;
        String newMarker = null;

        while (!done) {
            ListAccessKeysResponse response;

            if(newMarker == null) {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName).build();
                response = iam.listAccessKeys(request);
            } else {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .marker(newMarker).build();
                response = iam.listAccessKeys(request);
            }

            for (AccessKeyMetadata metadata :
                response.accessKeyMetadata()) {
                System.out.format("Retrieved access key %s",
                    metadata.accessKeyId());
            }

            if (!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }

    } catch (IamException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Hasil `listAccessKeys` paged (dengan maksimum default 100 record per panggilan). Anda dapat memanggil `isTruncated` [ListAccessKeysResponse](#) objek yang dikembalikan untuk melihat apakah kueri mengembalikan hasil yang lebih sedikit kemudian tersedia. Jika demikian, maka panggil `marker` `ListAccessKeysResponse` dan gunakan saat membuat permintaan baru. Gunakan permintaan baru itu dalam pemanggilan berikutnya. `listAccessKeys`

Lihat [contoh lengkapnya](#) di GitHub.

## Mengambil waktu terakhir yang digunakan kunci akses

Untuk mendapatkan waktu kunci akses terakhir digunakan, panggil `IamClient`'s `getAccessKeyLastUsed` metode dengan ID kunci akses (yang dapat diteruskan menggunakan [GetAccessKeyLastUsedRequest](#) objek).

Anda kemudian dapat menggunakan [GetAccessKeyLastUsedResponse](#) objek yang dikembalikan untuk mengambil waktu terakhir kunci yang digunakan.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetAccessKeyLastUsedRequest;
import software.amazon.awssdk.services.iam.model.GetAccessKeyLastUsedResponse;
import software.amazon.awssdk.services.iam.model.IamException;
```

### Kode

```
public static void getAccessKeyLastUsed(IamClient iam, String accessId ){

    try {
        GetAccessKeyLastUsedRequest request = GetAccessKeyLastUsedRequest.builder()
            .accessKeyId(accessId).build();

        GetAccessKeyLastUsedResponse response = iam.getAccessKeyLastUsed(request);

        System.out.println("Access key was last used at: " +
```

```
        response.accessKeyLastUsed().lastUsedDate());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Aktifkan atau nonaktifkan tombol akses

Anda dapat mengaktifkan atau menonaktifkan kunci akses dengan membuat [UpdateAccessKeyRequest](#) objek, memberikan ID kunci akses, opsional nama pengguna, dan yang diinginkan [status](#), lalu meneruskan objek permintaan ke metode `IamClient`'s `updateAccessKey`

### Impor

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

### Kode

```
public static void updateKey(IamClient iam, String username, String accessId,
String status ) {

    try {
        if (status.toLowerCase().equalsIgnoreCase("active")) {
            statusType = StatusType.ACTIVE;
        } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
            statusType = StatusType.INACTIVE;
        } else {
            statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
        }
        UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
            .accessKeyId(accessId)
            .userName(username)
```

```
        .status(statusType)
        .build();

iam.updateAccessKey(request);

System.out.printf(
    "Successfully updated the status of access key %s to" +
    "status %s for user %s", accessId, status, username);
} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Hapus kunci akses

Untuk menghapus kunci akses secara permanen, panggil `IamClient`'s `deleteKey` metode, berikan dengan ID dan nama pengguna kunci akses yang [DeleteAccessKeyRequest](#) berisi.

### Note

Setelah dihapus, kunci tidak dapat lagi diambil atau digunakan. Untuk menonaktifkan sementara kunci sehingga dapat diaktifkan lagi nanti, gunakan [updateAccessKey](#) metode sebagai gantinya.

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

## Kode

```
public static void deleteKey(IamClient iam ,String username, String accessKey ) {

    try {
```

```
DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
    .accessKeyId(accessKey)
    .userName(username)
    .build();

iam.deleteAccessKey(request);
System.out.println("Successfully deleted access key " + accessKey +
    " from user " + username);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Informasi lain

- [CreateAccessKey](#) di Referensi IAM API
- [ListAccessKeys](#) di Referensi IAM API
- [GetAccessKeyLastUsed](#) di Referensi IAM API
- [UpdateAccessKey](#) di Referensi IAM API
- [DeleteAccessKey](#) di Referensi IAM API

## Kelola IAM Pengguna

### Buat Pengguna

Buat IAM pengguna baru dengan memberikan nama pengguna ke `IamClient` `createUser` metode menggunakan [CreateUserRequest](#) objek yang berisi nama pengguna.

### Impor

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

```
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;
```

## Kode

```
public static String createIAMUser(IamClient iam, String username ) {

    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();

        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        CreateUserResponse response = iam.createUser(request);

        // Wait until the user is created
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user().userName();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Daftar Pengguna

Untuk membuat daftar IAM pengguna untuk akun Anda, buat yang baru [ListUsersRequest](#) dan teruskan ke `IamClient listUsers` metode ini. Anda dapat mengambil daftar pengguna dengan memanggil [ListUsersResponse](#) objek `users` yang dikembalikan.



Daftar pengguna yang dikembalikan `listUsers` oleh halaman. Anda dapat memeriksa untuk melihat ada lebih banyak hasil untuk diambil dengan memanggil `isTruncated` metode objek respons. Jika `isTruncated` mengembalikan `true`, maka panggil `marker()` metode objek respon. Gunakan nilai penanda untuk membuat objek permintaan baru. Kemudian panggil `listUsers` metode lagi dengan permintaan baru.

## Impor

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.services.iam.model.User;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

## Kode

```
public static void listAllUsers(IamClient iam ) {

    try {

        boolean done = false;
        String newMarker = null;

        while(!done) {
            ListUsersResponse response;

            if (newMarker == null) {
                ListUsersRequest request = ListUsersRequest.builder().build();
                response = iam.listUsers(request);
            } else {
                ListUsersRequest request = ListUsersRequest.builder()
                    .marker(newMarker).build();
                response = iam.listUsers(request);
            }

            for(User user : response.users()) {
                System.out.format("\n Retrieved user %s", user.userName());
            }

            if(!response.isTruncated()) {
                done = true;
            }
        }
    }
}
```

```
        } else {
            newMarker = response.marker();
        }
    }
} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Memperbarui Pengguna

Untuk memperbarui pengguna, panggil `updateUser` metode `IamClient` objek, yang mengambil [UpdateUserRequest](#) objek yang dapat Anda gunakan untuk mengubah nama atau jalur pengguna.

Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;
```

Kode

```
public static void updateIAMUser(IamClient iam, String curName, String newName) {

    try {
        UpdateUserRequest request = UpdateUserRequest.builder()
            .userName(curName)
            .newUserName(newName)
            .build();

        iam.updateUser(request);
        System.out.printf("Successfully updated user to username %s",
            newName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Menghapus Pengguna

Untuk menghapus pengguna, panggil `IamClient deleteUser` permintaan dengan [UpdateUserRequest](#) objek yang ditetapkan dengan nama pengguna untuk dihapus.

Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

Kode

```
public static void deleteIAMUser(IamClient iam, String userName) {

    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("Successfully deleted IAM user " + userName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Informasi Selengkapnya

- [IAM Pengguna](#) di Panduan IAM Pengguna
- [Mengelola IAM Pengguna](#) di Panduan IAM Pengguna
- [CreateUser](#) di Referensi IAM API
- [ListUsers](#) di Referensi IAM API
- [UpdateUser](#) di Referensi IAM API
- [DeleteUser](#) di Referensi IAM API

## Buat kebijakan IAM dengan AWS SDK for Java 2.x

[API Pembuat Kebijakan IAM](#) adalah pustaka yang dapat Anda gunakan untuk membangun [kebijakan IAM](#) di Java dan mengunggahnya ke AWS Identity and Access Management (IAM).

Alih-alih membangun kebijakan IAM dengan merakit string JSON secara manual atau dengan membaca file, API menyediakan pendekatan berorientasi objek sisi klien untuk menghasilkan string JSON. Saat Anda membaca kebijakan IAM yang ada dalam format JSON, API mengonversinya menjadi [IamPolicy](#) instance untuk ditangani.

API Pembuat Kebijakan IAM tersedia dengan SDK versi 2.20.105, jadi gunakan versi tersebut atau yang lebih baru di file build Maven Anda. Nomor versi terbaru SDK [terdaftar di pusat Maven](#).

Cuplikan berikut menunjukkan contoh blok ketergantungan untuk file Maven. `pom.xml` Ini memungkinkan Anda untuk menggunakan IAM Policy Builder API dalam proyek Anda.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>iam-policy-builder</artifactId>
  <version>2.20.139</version>
</dependency>
```

### Buat `IamPolicy`

Bagian ini menunjukkan beberapa contoh cara membuat kebijakan dengan menggunakan IAM Policy Builder API.

Dalam setiap contoh berikut, mulailah dengan [IamPolicy.Builder](#) dan tambahkan satu atau lebih pernyataan dengan menggunakan `addStatement` metode. Mengikuti pola ini, [IamStatement.Builder](#) memiliki metode untuk menambahkan efek, tindakan, sumber daya, dan kondisi ke pernyataan.

Contoh: Membuat kebijakan berbasis waktu

Contoh berikut membuat kebijakan berbasis identitas yang mengizinkan tindakan Amazon DynamoDB antara dua titik waktu `GetItem`.

```
public String timeBasedPolicyExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addResource(IamResource.ALL)
```

```

        .addCondition(b1 -> b1
            .operator(IamConditionOperator.DATE_GREATER_THAN)
            .key("aws:CurrentTime")
            .value("2020-04-01T00:00:00Z"))
        .addCondition(b1 -> b1
            .operator(IamConditionOperator.DATE_LESS_THAN)
            .key("aws:CurrentTime")
            .value("2020-06-30T23:59:59Z")))
    .build();

    // Use an IamPolicyWriter to write out the JSON string to a more readable
    format.
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true)
        .build());
}

```

## Keluaran JSON

Pernyataan terakhir dalam contoh sebelumnya mengembalikan string JSON berikut.

Baca lebih lanjut tentang [contoh](#) ini di Panduan AWS Identity and Access Management Pengguna.

```

{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : "dynamodb:GetItem",
    "Resource" : "*",
    "Condition" : {
      "DateGreaterThan" : {
        "aws:CurrentTime" : "2020-04-01T00:00:00Z"
      },
      "DateLessThan" : {
        "aws:CurrentTime" : "2020-06-30T23:59:59Z"
      }
    }
  }
}

```

## Contoh: Tentukan beberapa kondisi

Contoh berikut menunjukkan bagaimana Anda dapat membuat kebijakan berbasis identitas yang memungkinkan akses ke atribut DynamoDB tertentu. Kebijakan tersebut berisi dua syarat.

```

public String multipleConditionsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addAction("dynamodb:BatchGetItem")
            .addAction("dynamodb:Query")
            .addAction("dynamodb:PutItem")
            .addAction("dynamodb:UpdateItem")
            .addAction("dynamodb>DeleteItem")
            .addAction("dynamodb:BatchWriteItem")
            .addResource("arn:aws:dynamodb:*:*:table/table-name")

        .addConditions(IamConditionOperator.STRING_EQUALS.addPrefix("ForAllValues:"),
            "dynamodb:Attributes",
            List.of("column-name1", "column-name2", "column-
name3"))

            .addCondition(b1 ->
b1.operator(IamConditionOperator.STRING_EQUALS.addSuffix("IfExists"))
                .key("dynamodb>Select")
                .value("SPECIFIC_ATTRIBUTES"))

        .build();

    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

## Keluaran JSON

Pernyataan terakhir dalam contoh sebelumnya mengembalikan string JSON berikut.

Baca lebih lanjut tentang [contoh](#) ini di Panduan AWS Identity and Access Management Pengguna.

```

{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : [ "dynamodb:GetItem", "dynamodb:BatchGetItem", "dynamodb:Query",
"dynamodb:PutItem", "dynamodb:UpdateItem", "dynamodb>DeleteItem",
"dynamodb:BatchWriteItem" ],
    "Resource" : "arn:aws:dynamodb:*:*:table/table-name",
    "Condition" : {
      "ForAllValues:StringEquals" : {

```

```

    "dynamodb:Attributes" : [ "column-name1", "column-name2", "column-name3" ]
  },
  "StringEqualsIfExists" : {
    "dynamodb:Select" : "SPECIFIC_ATTRIBUTES"
  }
}
}

```

Contoh: Tentukan prinsipal

Contoh berikut menunjukkan cara membuat kebijakan berbasis sumber daya yang menolak akses ke bucket untuk semua prinsipal kecuali yang ditentukan dalam kondisi.

```

public String specifyPrincipalsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.DENY)
            .addAction("s3:*")
            .addPrincipal(IamPrincipal.ALL)
            .addResource("arn:aws:s3:::BUCKETNAME/*")
            .addResource("arn:aws:s3:::BUCKETNAME")
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.ARN_NOT_EQUALS)
                .key("aws:PrincipalArn")
                .value("arn:aws:iam::444455556666:user/user-name")))
        .build();
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

Keluaran JSON

Pernyataan terakhir dalam contoh sebelumnya mengembalikan string JSON berikut.

Baca lebih lanjut tentang [contoh](#) ini di Panduan AWS Identity and Access Management Pengguna.

```

{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Deny",
    "Principal" : "*",
    "Action" : "s3:*",

```

```

"Resource" : [ "arn:aws:s3:::BUCKETNAME/*", "arn:aws:s3:::BUCKETNAME" ],
"Condition" : {
  "ArnNotEquals" : {
    "aws:PrincipalArn" : "arn:aws:iam::444455556666:user/user-name"
  }
}
}
}
}

```

Contoh: Izinkan akses lintas akun

Contoh berikut menunjukkan cara mengizinkan orang lain Akun AWS mengunggah objek ke bucket Anda sambil mempertahankan kontrol pemilik penuh atas objek yang diunggah.

```

public String allowCrossAccountAccessExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addPrincipal(IamPrincipalType.AWS, "111122223333")
            .addAction("s3:PutObject")
            .addResource("arn:aws:s3:::DOC-EXAMPLE-BUCKET/*")
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.STRING_EQUALS)
                .key("s3:x-amz-acl")
                .value("bucket-owner-full-control")))
        .build();
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

Keluaran JSON

Pernyataan terakhir dalam contoh sebelumnya mengembalikan string JSON berikut.

Baca lebih lanjut tentang [contoh](#) ini di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

```

{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Principal" : {

```



```

    "AWS" : "111122223333"
  },
  "Action" : "s3:PutObject",
  "Resource" : "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
  "Condition" : {
    "StringEquals" : {
      "s3:x-amz-acl" : "bucket-owner-full-control"
    }
  }
}
}
}
}

```

## Gunakan **IamPolicy** dengan IAM

Setelah Anda membuat `IamPolicy` instance, Anda menggunakan [IamClient](#) untuk bekerja dengan layanan IAM.

Contoh berikut membangun kebijakan yang memungkinkan [identitas IAM](#) untuk menulis item ke tabel DynamoDB di akun yang ditentukan dengan parameter. `accountID` Kebijakan tersebut kemudian diunggah ke IAM sebagai string JSON.

```

public String createAndUploadPolicyExample(IamClient iam, String accountID, String
policyName) {
    // Build the policy.
    IamPolicy policy =
        IamPolicy.builder() // 'version' defaults to "2012-10-17".
            .addStatement(IamStatement.builder()
                .effect(IamEffect.ALLOW)
                .addAction("dynamodb:PutItem")
                .addResource("arn:aws:dynamodb:us-east-1:" + accountID
+ ":table/exampleTableName")
                .build())
            .build();
    // Upload the policy.
    iam.createPolicy(r ->
r.policyName(policyName).policyDocument(policy.toJson()));
    return policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
}

```

Contoh berikutnya dibangun di atas contoh sebelumnya. Kode mengunduh kebijakan dan menggunakannya sebagai dasar untuk kebijakan baru dengan menyalin dan mengubah pernyataan. Kebijakan baru kemudian diunggah.

```

public String createNewBasedOnExistingPolicyExample(IamClient iam, String
accountID, String policyName, String newPolicyName) {

    String policyArn = "arn:aws:iam::" + accountID + ":policy/" + policyName;
    GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

    String policyVersion = getPolicyResponse.policy().defaultVersionId();
    GetPolicyVersionResponse getPolicyVersionResponse =
        iam.getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

    // Create an IamPolicy instance from the JSON string returned from IAM.
    String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
StandardCharsets.UTF_8);
    IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

    /*
    All IamPolicy components are immutable, so use the copy method that
creates a new instance that
    can be altered in the same method call.

    Add the ability to get an item from DynamoDB as an additional action.
    */
    IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

    // Create a new statement that replaces the original statement.
    IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

    // Upload the new policy. IAM now has both policies.
    iam.createPolicy(r -> r.policyName(newPolicyName)
        .policyDocument(newPolicy.toJson()));

    return newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
}

```

## IamClient

Contoh sebelumnya menggunakan IamClient argumen yang dibuat seperti yang ditunjukkan pada cuplikan berikut.

```
IamClient iam = IamClient.builder().region(Region.AWS_GLOBAL).build();
```

## Kebijakan di JSON

Contoh mengembalikan string JSON berikut.

First example

```
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : "dynamodb:PutItem",
    "Resource" : "arn:aws:dynamodb:us-east-1:111122223333:table/exampleTableName"
  }
}
```

Second example

```
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Action" : [ "dynamodb:PutItem", "dynamodb:GetItem" ],
    "Resource" : "arn:aws:dynamodb:us-east-1:111122223333:table/exampleTableName"
  }
}
```

## Bekerja dengan IAM kebijakan

### Buat kebijakan

Untuk membuat kebijakan baru, berikan nama kebijakan dan dokumen kebijakan berformat JSON dalam metode [CreatePolicyRequest](#) to the's. IamClient createPolicy

### Impor

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
```

## Kode

```
public static String createIAMPolicy(IamClient iam, String policyName ) {

    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();

        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument).build();

        CreatePolicyResponse response = iam.createPolicy(request);

        // Wait until the policy is created
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);
        waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Dapatkan kebijakan

Untuk mengambil kebijakan yang ada, panggil `getPolicy` metode, yang menyediakan ARN kebijakan dalam [GetPolicyRequest](#) objek. `IamClient`

## Impor

```
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

## Kode

```
public static void getIAMPolicy(IamClient iam, String policyArn) {

    try {
        GetPolicyRequest request = GetPolicyRequest.builder()
            .policyArn(policyArn).build();

        GetPolicyResponse response = iam.getPolicy(request);
        System.out.format("Successfully retrieved policy %s",
            response.policy().policyName());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Lampirkan kebijakan peran

Anda dapat melampirkan kebijakan ke IAM [peran](#) dengan memanggil `attachRolePolicy` metode, memberikannya nama peran dan kebijakan ARN di `IamClient` [AttachRolePolicyRequest](#)

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;
```

## Kode

```
public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {

    try {

        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
        .roleName(roleName)
        .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        // Ensure that the policy is not attached to this role
        String polArn = "";
        for (AttachedPolicy policy: attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn)==0) {
                System.out.println(roleName +
                    " policy is already attached to this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
            AttachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policyArn)
                .build();

        iam.attachRolePolicy(attachRequest);

        System.out.println("Successfully attached policy " + policyArn +
            " to role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("Done");
}
```

```
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Daftar kebijakan peran terlampir

Buat daftar kebijakan terlampir pada peran dengan memanggil `listAttachedRolePolicies` metode ini. `IamClient` Dibutuhkan [ListAttachedRolePoliciesRequest](#) objek yang berisi nama peran untuk mencantumkan kebijakan.

Panggil `getAttachedPolicies` [ListAttachedRolePoliciesResponse](#) objek yang dikembalikan untuk mendapatkan daftar kebijakan terlampir. Hasil dapat terpotong; jika `isTruncated` metode `ListAttachedRolePoliciesResponse` objek kembali `true`, panggil metode `ListAttachedRolePoliciesResponse` objek. `marker` Gunakan penanda yang dikembalikan untuk membuat permintaan baru dan menggunakannya untuk memanggil `listAttachedRolePolicies` lagi untuk mendapatkan kumpulan hasil berikutnya.

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;
```

## Kode

```
public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {

    try {

        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                .roleName(roleName)
                .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
```

```
List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

// Ensure that the policy is not attached to this role
String polArn = "";
for (AttachedPolicy policy: attachedPolicies) {
    polArn = policy.policyArn();
    if (polArn.compareTo(policyArn)==0) {
        System.out.println(roleName +
            " policy is already attached to this role.");
        return;
    }
}

AttachRolePolicyRequest attachRequest =
    AttachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(policyArn)
        .build();

iam.attachRolePolicy(attachRequest);

System.out.println("Successfully attached policy " + policyArn +
    " to role " + roleName);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Done");
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Lepaskan kebijakan peran

Untuk melepaskan kebijakan dari peran, panggil `detachRolePolicy` metode `IamClient` ini, berikan nama peran dan kebijakan ARN dalam file. [DetachRolePolicyRequest](#)

### Impor

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
```



```
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

## Kode

```
public static void detachPolicy(IamClient iam, String roleName, String policyArn )
{
    try {
        DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.detachRolePolicy(request);
        System.out.println("Successfully detached policy " + policyArn +
            " from role " + roleName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Informasi lain

- [Ikhtisar IAM Kebijakan](#) dalam Panduan IAM Pengguna.
- [AWSReferensi Kebijakan IAM](#) dalam Panduan IAM Pengguna.
- [CreatePolicy](#) di Referensi IAM API
- [GetPolicy](#) di Referensi IAM API
- [AttachRolePolicy](#) di Referensi IAM API
- [ListAttachedRolePolicies](#) di Referensi IAM API
- [DetachRolePolicy](#) di Referensi IAM API

## Bekerja dengan sertifikat IAM server

Untuk mengaktifkan koneksi HTTPS ke situs web atau aplikasi Anda AWS, Anda memerlukan sertifikat server SSL/TLS. Anda dapat menggunakan sertifikat server yang disediakan oleh AWS Certificate Manager atau yang Anda peroleh dari penyedia eksternal.

Sebaiknya gunakan ACM untuk menyediakan, mengelola, dan menyebarkan sertifikat server Anda. Dengan ACM Anda dapat meminta sertifikat, menyebarkannya ke AWS sumber daya Anda, dan membiarkan ACM menangani perpanjangan sertifikat untuk Anda. Sertifikat yang ACM disediakan oleh gratis. Untuk informasi selengkapnya ACM, lihat [Panduan AWS Certificate Manager Pengguna](#).

### Dapatkan sertifikat server

Anda dapat mengambil sertifikat server dengan memanggil `getServerCertificate` metode, meneruskannya [GetServerCertificateRequest](#) dengan nama sertifikat. `IamClient`

#### Impor

```
import software.amazon.awssdk.services.iam.model.GetServerCertificateRequest;
import software.amazon.awssdk.services.iam.model.GetServerCertificateResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

#### Kode

```
public static void getCertificate(IamClient iam,String certName ) {

    try {
        GetServerCertificateRequest request = GetServerCertificateRequest.builder()
            .serverCertificateName(certName)
            .build();

        GetServerCertificateResponse response = iam.getServerCertificate(request);
        System.out.format("Successfully retrieved certificate with body %s",
            response.getServerCertificate().certificateBody());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Daftar sertifikat server

Untuk mencantumkan sertifikat server Anda, panggil `listServerCertificates` metode ini dengan file [ListServerCertificatesRequest](#). `IamClient` ini mengembalikan a [ListServerCertificatesResponse](#).

Panggil `serverCertificateMetadataList` metode `ListServerCertificateResponse` objek yang dikembalikan untuk mendapatkan daftar [ServerCertificateMetadata](#) objek yang dapat Anda gunakan untuk mendapatkan informasi tentang setiap sertifikat.

Hasil dapat terpotong; jika `isTruncated` metode `ListServerCertificateResponse` objek kembali `true`, panggil `marker` metode `ListServerCertificatesResponse` objek dan gunakan penanda untuk membuat permintaan baru. Gunakan permintaan baru untuk menelepon `listServerCertificates` lagi untuk mendapatkan kumpulan hasil berikutnya.

## Impor

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListServerCertificatesRequest;
import software.amazon.awssdk.services.iam.model.ListServerCertificatesResponse;
import software.amazon.awssdk.services.iam.model.ServerCertificateMetadata;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

## Kode

```
public static void listCertificates(IamClient iam) {

    try {
        boolean done = false;
        String newMarker = null;

        while(!done) {
            ListServerCertificatesResponse response;

            if (newMarker == null) {
                ListServerCertificatesRequest request =
```

```
        ListServerCertificatesRequest.builder().build());
        response = iam.listServerCertificates(request);
    } else {
        ListServerCertificatesRequest request =
            ListServerCertificatesRequest.builder()
                .marker(newMarker).build();
        response = iam.listServerCertificates(request);
    }

    for(ServerCertificateMetadata metadata :
        response.serverCertificateMetadataList()) {
        System.out.printf("Retrieved server certificate %s",
            metadata.serverCertificateName());
    }

    if(!response.isTruncated()) {
        done = true;
    } else {
        newMarker = response.marker();
    }
}

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Memperbarui sertifikat server

Anda dapat memperbarui nama atau jalur sertifikat server dengan memanggil `updateServerCertificate` metode ini. `IamClient` Dibutuhkan [UpdateServerCertificateRequest](#) objek yang ditetapkan dengan nama sertifikat server saat ini dan baik nama baru atau jalur baru untuk digunakan.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateServerCertificateRequest;
```

```
import software.amazon.awssdk.services.iam.model.UpdateServerCertificateResponse;
```

## Kode

```
public static void updateCertificate(IamClient iam, String curName, String newName)
{
    try {
        UpdateServerCertificateRequest request =
            UpdateServerCertificateRequest.builder()
                .serverCertificateName(curName)
                .newServerCertificateName(newName)
                .build();

        UpdateServerCertificateResponse response =
            iam.updateServerCertificate(request);

        System.out.printf("Successfully updated server certificate to name %s",
            newName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Hapus sertifikat server

Untuk menghapus sertifikat server, panggil `deleteServerCertificate` metode dengan [DeleteServerCertificateRequest](#) berisi nama sertifikat. `IamClient`

## Impor

```
import software.amazon.awssdk.services.iam.model.DeleteServerCertificateRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

## Kode

```
public static void deleteCert(IamClient iam,String certName ) {  
  
    try {  
        DeleteServerCertificateRequest request =  
            DeleteServerCertificateRequest.builder()  
                .serverCertificateName(certName)  
                .build();  
  
        iam.deleteServerCertificate(request);  
        System.out.println("Successfully deleted server certificate " +  
            certName);  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Informasi lain

- [Bekerja dengan Sertifikat Server](#) di Panduan IAM Pengguna
- [GetServerCertificate](#) di Referensi IAM API
- [ListServerCertificates](#) di Referensi IAM API
- [UpdateServerCertificate](#) di Referensi IAM API
- [DeleteServerCertificate](#) di Referensi IAM API
- [AWS Certificate Manager Panduan Pengguna](#)

## Bekerja dengan Kinesis

Bagian ini memberikan contoh pemrograman [Amazon Kinesis](#) menggunakan AWS SDK for Java 2.x.

Untuk informasi selengkapnya Kinesis, lihat [Panduan Amazon Kinesis Pengembang](#).

Contoh berikut hanya mencakup kode yang diperlukan untuk mendemonstrasikan setiap teknik. [Kode contoh lengkap tersedia di GitHub](#). Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk dibangun dan dijalankan.

## Topik

- [Berlangganan Amazon Kinesis Data Streams](#)

## Berlangganan Amazon Kinesis Data Streams

Contoh berikut menunjukkan cara mengambil dan memproses data dari Amazon Kinesis Data Streams menggunakan metode `inSubscribeToShard`. Kinesis Data Streams sekarang menggunakan fitur fanout yang disempurnakan dan API pengambilan data HTTP/2 latensi rendah, sehingga memudahkan pengembang untuk menjalankan beberapa aplikasi latensi rendah dan berkinerja tinggi pada Data Stream yang sama. Kinesis

### Penyiapan

Pertama, buat Kinesis klien asinkron dan objek [SubscribeToShardRequest](#). Objek-objek ini digunakan dalam masing-masing contoh berikut untuk berlangganan Kinesis acara.

### Impor

```
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.atomic.AtomicInteger;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;
import software.amazon.awssdk.services.kinesis.model.ShardIteratorType;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEvent;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEventStream;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardRequest;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponse;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponseHandler;
```

### Kode

```
Region region = Region.US_EAST_1;
KinesisAsyncClient client = KinesisAsyncClient.builder()
    .region(region)
    .build();

SubscribeToShardRequest request = SubscribeToShardRequest.builder()
    .consumerARN(CONSUMER_ARN)
```

```

        .shardId("arn:aws:kinesis:us-east-1:111122223333:stream/
StockTradeStream")
        .startingPosition(s -> s.type(ShardIteratorType.LATEST)).build();

```

## Gunakan antarmuka pembangun

Anda dapat menggunakan builder metode ini untuk menyederhanakan pembuatan.

### [SubscribeToShardResponseHandler](#)

Dengan menggunakan builder, Anda dapat mengatur setiap panggilan balik siklus hidup dengan panggilan metode alih-alih mengimplementasikan antarmuka lengkap.

#### Kode

```

private static CompletableFuture<Void> responseHandlerBuilder(KinesisAsyncClient
client, SubscribeToShardRequest request) {
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
        .builder()
        .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
        .onComplete(() -> System.out.println("All records stream
successfully"))
        // Must supply some type of subscriber
        .subscriber(e -> System.out.println("Received event - " + e))
        .build();
    return client.subscribeToShard(request, responseHandler);
}

```

Untuk kontrol lebih lanjut dari penerbit, Anda dapat menggunakan `publisherTransformer` metode untuk menyesuaikan penerbit.

#### Kode

```

private static CompletableFuture<Void>
responseHandlerBuilderPublisherTransformer(KinesisAsyncClient client,
SubscribeToShardRequest request) {
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
        .builder()
        .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))

```



```

        .publisherTransformer(p -> p.filter(e -> e instanceof
SubscribeToShardEvent).limit(100))
        .subscriber(e -> System.out.println("Received event - " + e))
        .build();
    return client.subscribeToShard(request, responseHandler);
}

```

Lihat [contoh lengkapnya](#) di GitHub.

## Gunakan handler respon kustom

Untuk kontrol penuh pelanggan dan penerbit, terapkan antarmuka.

### SubscribeToShardResponseHandler

Dalam contoh ini, Anda menerapkan `onEventStream` metode, yang memungkinkan Anda akses penuh ke penerbit. Ini menunjukkan bagaimana mengubah penerbit menjadi catatan peristiwa untuk dicetak oleh pelanggan.

### Kode

```

    private static CompletableFuture<Void>
responseHandlerBuilderClassic(KinesisAsyncClient client, SubscribeToShardRequest
request) {
    SubscribeToShardResponseHandler responseHandler = new
SubscribeToShardResponseHandler() {

        @Override
        public void responseReceived(SubscribeToShardResponse response) {
            System.out.println("Receieved initial response");
        }

        @Override
        public void onEventStream(SdkPublisher<SubscribeToShardEventStream>
publisher) {
            publisher
                // Filter to only SubscribeToShardEvents
                .filter(SubscribeToShardEvent.class)
                // Flat map into a publisher of just records
                .flatMapIterable(SubscribeToShardEvent::records)
                // Limit to 1000 total records
                .limit(1000)
                // Batch records into lists of 25
                .buffer(25)

```

```

        // Print out each record batch
        .subscribe(batch -> System.out.println("Record Batch - " +
batch));
    }

    @Override
    public void complete() {
        System.out.println("All records stream successfully");
    }

    @Override
    public void exceptionOccurred(Throwable throwable) {
        System.err.println("Error during stream - " + throwable.getMessage());
    }
};
return client.subscribeToShard(request, responseHandler);
}

```

Lihat [contoh lengkapnya](#) di GitHub.

## Gunakan antarmuka pengunjung

Anda dapat menggunakan objek [Pengunjung](#) untuk berlangganan acara tertentu yang ingin Anda tonton.

### Kode

```

private static CompletableFuture<Void>
responseHandlerBuilderVisitorBuilder(KinesisAsyncClient client,
SubscribeToShardRequest request) {
    SubscribeToShardResponseHandler.Visitor visitor =
SubscribeToShardResponseHandler.Visitor
        .builder()
        .onSubscribeToShardEvent(e -> System.out.println("Received subscribe to
shard event " + e))
        .build();
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
        .builder()
        .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
        .subscriber(visitor)
        .build();
}

```

```
    return client.subscribeToShard(request, responseHandler);
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Gunakan pelanggan khusus

Anda juga dapat menerapkan pelanggan khusus Anda sendiri untuk berlangganan streaming.

Cuplikan kode ini menunjukkan contoh pelanggan.

### Kode

```
private static class MySubscriber implements
Subscriber<SubscribeToShardEventStream> {

    private Subscription subscription;
    private AtomicInteger eventCount = new AtomicInteger(0);

    @Override
    public void onSubscribe(Subscription subscription) {
        this.subscription = subscription;
        this.subscription.request(1);
    }

    @Override
    public void onNext(SubscribeToShardEventStream shardSubscriptionEventStream) {
        System.out.println("Received event " + shardSubscriptionEventStream);
        if (eventCount.incrementAndGet() >= 100) {
            // You can cancel the subscription at any time if you wish to stop
receiving events.
            subscription.cancel();
        }
        subscription.request(1);
    }

    @Override
    public void onError(Throwable throwable) {
        System.err.println("Error occurred while stream - " +
throwable.getMessage());
    }

    @Override
```

```
public void onComplete() {
    System.out.println("Finished streaming all events");
}
}
```

Anda dapat meneruskan pelanggan kustom ke `subscribe` metode seperti yang ditunjukkan pada cuplikan kode berikut.

## Kode

```
private static CompletableFuture<Void>
responseHandlerBuilderSubscriber(KinesisAsyncClient client, SubscribeToShardRequest
request) {
    SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
        .builder()
        .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
        .subscriber(MySubscriber::new)
        .build();
    return client.subscribeToShard(request, responseHandler);
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Menulis catatan data ke dalam aliran Kinesis data

Anda dapat menggunakan [KinesisClient](#) objek untuk menulis catatan data ke dalam aliran Kinesis data dengan menggunakan `putRecords` metode ini. Untuk berhasil memanggil metode ini, buat [PutRecordsRequest](#) objek. Anda meneruskan nama aliran data ke `streamName` metode. Anda juga harus meneruskan data dengan menggunakan `putRecords` metode (seperti yang ditunjukkan pada contoh kode berikut).

## Impor

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
```

```
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;
```

Dalam contoh kode Java berikut, perhatikan bahwa `StockTrade` objek digunakan sebagai data untuk menulis ke aliran Kinesis data. Sebelum menjalankan contoh ini, pastikan Anda telah membuat aliran data.

## Kode

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StockTradesWriter {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <streamName>

                Where:
                streamName - The Amazon Kinesis data stream to which records are
                written (for example, StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
```

```
KinesisClient kinesisClient = KinesisClient.builder()
    .region(region)
    .build();

// Ensure that the Kinesis Stream is valid.
validateStream(kinesisClient, streamName);
setStockData(kinesisClient, streamName);
kinesisClient.close();
}

public static void setStockData(KinesisClient kinesisClient, String streamName) {
    try {
        // Repeatedly send stock trades with a 100 milliseconds wait in between.
        StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

        // Put in 50 Records for this example.
        int index = 50;
        for (int x = 0; x < index; x++) {
            StockTrade trade = stockTradeGenerator.getRandomTrade();
            sendStockTrade(trade, kinesisClient, streamName);
            Thread.sleep(100);
        }

    } catch (KinesisException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done");
}

private static void sendStockTrade(StockTrade trade, KinesisClient kinesisClient,
    String streamName) {
    byte[] bytes = trade.toJsonAsBytes();

    // The bytes could be null if there is an issue with the JSON serialization by
    // the Jackson JSON library.
    if (bytes == null) {
        System.out.println("Could not get JSON bytes for stock trade");
        return;
    }

    System.out.println("Putting trade: " + trade);
    PutRecordRequest request = PutRecordRequest.builder()
```

```
        .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol as
the partition key, explained in
                                                // the Supplemental Information
section below.
        .streamName(streamName)
        .data(SdkBytes.fromByteArray(bytes))
        .build();

    try {
        kinesisClient.putRecord(request);
    } catch (KinesisException e) {
        System.err.println(e.getMessage());
    }
}

private static void validateStream(KinesisClient kinesisClient, String streamName)
{
    try {
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
            .streamName(streamName)
            .build();

        DescribeStreamResponse describeStreamResponse =
kinesisClient.describeStream(describeStreamRequest);

        if (!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
        {
            System.err.println("Stream " + streamName + " is not active. Please
wait a few moments and try again.");
            System.exit(1);
        }

    } catch (KinesisException e) {
        System.err.println("Error found while describing the stream " +
streamName);
        System.err.println(e);
        System.exit(1);
    }
}
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Gunakan pustaka pihak ketiga

Anda dapat menggunakan pustaka pihak ketiga lainnya alih-alih menerapkan pelanggan khusus. Contoh ini menunjukkan penggunaan RxJava implementasi, tetapi Anda dapat menggunakan pustaka apa pun yang mengimplementasikan antarmuka Reactive Streams. Lihat [halaman RxJava wiki di Github](#) untuk informasi lebih lanjut tentang perpustakaan itu.

Untuk menggunakan perpustakaan, tambahkan sebagai dependensi. Jika Anda menggunakan Maven, contoh menunjukkan cuplikan POM untuk digunakan.

### Entri POM

```
<dependency>
  <groupId>io.reactivex.rxjava2</groupId>
  <artifactId>rxjava</artifactId>
  <version>2.1.14</version>
</dependency>
```

### Impor

```
import java.net.URI;
import java.util.concurrent.CompletableFuture;

import io.reactivex.Flowable;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.http.Protocol;
import software.amazon.awssdk.http.SdkHttpConfigurationOption;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;
import software.amazon.awssdk.services.kinesis.model.ShardIteratorType;
import software.amazon.awssdk.services.kinesis.model.StartingPosition;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEvent;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardRequest;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponseHandler;
import software.amazon.awssdk.utils.AttributeMap;
```

Contoh ini digunakan RxJava dalam metode `onEventStream` siklus hidup. Ini memberi Anda akses penuh ke penerbit, yang dapat digunakan untuk membuat Rx Flowable.



## Kode

```

SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .onEventStream(p -> Flowable.fromPublisher(p)
        .ofType(SubscribeToShardEvent.class)

.flatMapIterable(SubscribeToShardEvent::records)
        .limit(1000)
        .buffer(25)
        .subscribe(e -> System.out.println("Record
batch = " + e)))
    .build();

```

Anda juga dapat menggunakan `publisherTransformer` metode ini dengan `Flowable` penerbit. Anda harus menyesuaikan `Flowable` penerbit dengan `SdkPublisher`, seperti yang ditunjukkan pada contoh berikut.

## Kode

```

SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .publisherTransformer(p ->
SdkPublisher.adapt(Flowable.fromPublisher(p).limit(100)))
    .build();

```

Lihat [contoh lengkapnya](#) di GitHub.

## Informasi lain

- [SubscribeToShardEvent](#) di Referensi Amazon Kinesis API
- [SubscribeToShard](#) di Referensi Amazon Kinesis API

# Memanggil, membuat daftar, dan menghapus fungsi AWS Lambda

Bagian ini memberikan contoh pemrograman dengan klien Lambda layanan dengan menggunakan AWS SDK for Java 2.x.

Topik

- [Memanggil fungsi Lambda](#)
- [Daftar fungsi Lambda](#)
- [Hapus fungsi Lambda](#)

## Memanggil fungsi Lambda

Anda dapat memanggil Lambda fungsi dengan membuat [LambdaClient](#) objek dan menjalankan metodenya `invoke`. Buat [InvokeRequest](#) objek untuk menentukan informasi tambahan seperti nama fungsi dan payload untuk diteruskan ke Lambda fungsi. Nama fungsi muncul sebagai `arn:aws:lambda: us-east- 1:123456789012: function:. HelloFunction` Anda dapat mengambil nilai dengan melihat fungsi di. AWS Management Console

Untuk meneruskan data payload ke suatu fungsi, buat [SdkBytes](#) objek yang berisi informasi. Misalnya, dalam contoh kode berikut, perhatikan data JSON diteruskan ke Lambda fungsi.

Impor

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.InvokeRequest;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lambda.model.InvokeResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;
```

Kode

Contoh kode berikut menunjukkan bagaimana untuk memanggil fungsi. Lambda

```
public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res = null ;
    try {
```

```
//Need a SdkBytes instance for the payload
String json = "{\"Hello \": \"Paris\"}";
SdkBytes payload = SdkBytes.fromUtf8String(json) ;

//Setup an InvokeRequest
InvokeRequest request = InvokeRequest.builder()
    .functionName(functionName)
    .payload(payload)
    .build();

res = awsLambda.invoke(request);
String value = res.payload().asUtf8String() ;
System.out.println(value);

} catch(LambdaException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Daftar fungsi Lambda

Bangun [Lambda Client](#) objek dan panggil `listFunctions` metodenya. Metode ini mengembalikan [ListFunctionsResponse](#) objek. Anda dapat memanggil `functions` metode objek ini untuk mengembalikan daftar [FunctionConfiguration](#) objek. Anda dapat mengulangi melalui daftar untuk mengambil informasi tentang fungsi. Misalnya, contoh kode Java berikut menunjukkan cara mendapatkan setiap nama fungsi.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.ListFunctionsResponse;
import software.amazon.awssdk.services.lambda.model.FunctionConfiguration;
import java.util.List;
```

### Kode

Contoh kode Java berikut menunjukkan bagaimana untuk mengambil daftar nama fungsi.

```
public static void listFunctions(LambdaClient awsLambda) {  
  
    try {  
        ListFunctionsResponse functionResult = awsLambda.listFunctions();  
        List<FunctionConfiguration> list = functionResult.functions();  
  
        for (FunctionConfiguration config: list) {  
            System.out.println("The function name is "+config.functionName());  
        }  
  
    } catch(LambdaException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Hapus fungsi Lambda

Bangun [LambdaClient](#) objek dan panggil `deleteFunction` metodenya. Buat [DeleteFunctionRequest](#) objek dan berikan ke `deleteFunction` metode. Objek ini berisi informasi seperti nama fungsi yang akan dihapus. Nama fungsi muncul sebagai `arn:aws:lambda:us-east-1:123456789012:function:. HelloFunction` Anda dapat mengambil nilai dengan melihat fungsi di AWS Management Console

Impor

```
import software.amazon.awssdk.services.lambda.LambdaClient;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.lambda.model.DeleteFunctionRequest;  
import software.amazon.awssdk.services.lambda.model.LambdaException;
```

Kode

Kode Java berikut menunjukkan cara menghapus Lambda fungsi.

```
public static void deleteLambdaFunction(LambdaClient awsLambda, String  
functionName ) {  
    try {  
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()  
            .functionName(functionName)
```

```
        .build();

        awsLambda.deleteFunction(request);
        System.out.println("The "+functionName +" function was deleted");

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Bekerja dengan Amazon S3

Bagian ini memberikan contoh pemrograman dengan [Amazon Simple Storage Service \(S3\) menggunakan file](#). AWS SDK for Java 2.x

Contoh berikut hanya mencakup kode yang diperlukan untuk mendemonstrasikan setiap teknik. [Kode contoh lengkap tersedia di GitHub](#). Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk dibangun dan dijalankan.

### Note

Dari versi 2.18.x dan seterusnya, AWS SDK for Java 2.x menggunakan pengalamatan gaya host virtual saat menyertakan penggantian titik akhir. Ini berlaku selama nama bucket adalah label DNS yang valid.

Panggil [forcePathStyle](#) metode dengan `true` pembuat klien Anda untuk memaksa klien menggunakan pengalamatan gaya jalur untuk bucket.

Contoh berikut menunjukkan klien layanan yang dikonfigurasi dengan penggantian titik akhir dan menggunakan pengalamatan gaya jalur.

```
S3Client client = S3Client.builder()
    .region(Region.US_WEST_2)
    .endpointOverride(URI.create("https://s3.us-
west-2.amazonaws.com"))
    .forcePathStyle(true)
    .build();
```

## Gunakan titik akses atau Titik Akses Multi-Wilayah

Setelah [jalur akses Amazon S3](#) atau [Titik Akses Multi-Wilayah](#) disiapkan, Anda dapat memanggil metode objek, seperti `putObject` dan `getObject` dan memberikan pengenal titik akses alih-alih nama bucket.

Misalnya, jika pengidentifikasi ARN titik akses adalah `arn:aws:s3:us-west-2:123456789012:accesspoint/test`, Anda dapat menggunakan cuplikan berikut untuk memanggil metode `putObject`

```
Path path = Paths.get(URI.create("file:///temp/file.txt"));

s3Client.putObject(builder -> builder
    .key("myKey")
    .bucket("arn:aws:s3:us-west-2:123456789012:accesspoint/test")
    , path);
```

Di tempat string ARN, Anda juga dapat menggunakan [alias gaya ember dari titik](#) akses untuk parameter `bucket`

Untuk menggunakan Multi-Region Access Point, ganti `bucket` parameter dengan Multi-Region Access Point ARN yang memiliki format berikut.

```
arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias
```

Tambahkan dependensi Maven berikut untuk bekerja dengan Multi-Region Access Points menggunakan SDK for Java. Cari maven central untuk [versi terbaru](#).

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>auth-crt</artifactId>
  <version>VERSION</version>
</dependency>
```

### Topik

- [Buat, daftar, dan hapus Amazon S3 ember](#)
- [Bekerja dengan Amazon S3 benda](#)
- [Bekerja dengan URL yang Amazon S3 telah ditandatangani sebelumnya](#)

- [Akses Lintas Wilayah untuk Amazon S3](#)
- [Gunakan klien S3 berkinerja: klien S3 berbasis AWS CRT](#)
- [Transfer file dan direktori dengan Amazon S3 Transfer Manager](#)

## Buat, daftar, dan hapus Amazon S3 ember

Setiap objek (file) di Amazon S3 harus berada dalam ember. Sebuah ember mewakili koleksi (wadah) objek. Setiap ember harus memiliki kunci (nama) yang unik. Untuk informasi terperinci tentang bucket dan konfigurasinya, lihat [Bekerja dengan Amazon S3 Bucket](#) di Amazon Simple Storage Service Panduan Pengguna.

### Note

#### Praktik Terbaik

Kami menyarankan Anda mengaktifkan aturan [AbortIncompleteMultipartUpload](#) siklus hidup pada bucket Anda Amazon S3.

Aturan ini mengarahkan Amazon S3 untuk membatalkan unggahan multipart yang tidak selesai dalam jumlah hari tertentu setelah dimulai. Ketika batas waktu yang ditetapkan terlampaui, Amazon S3 batalkan unggahan dan kemudian menghapus data unggahan yang tidak lengkap.

Untuk informasi selengkapnya, lihat [Konfigurasi Siklus Hidup untuk Bucket dengan Pembuatan Versi di Panduan Pengguna](#). Amazon Simple Storage Service

### Note

Cuplikan kode ini mengasumsikan bahwa Anda memahami materi dalam dasar-dasar, dan telah mengonfigurasi AWS kredensi default menggunakan informasi di [the section called "Menyiapkan akses masuk tunggal untuk SDK"](#)

## Buat bucket

Buat [CreateBucketRequest](#) dan berikan nama ember. Teruskan ke metode `S3Client.createBucket`. Gunakan `S3Client` untuk melakukan operasi tambahan seperti mencantumkan atau menghapus bucket seperti yang ditunjukkan pada contoh selanjutnya.

## Impor

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

## Kode

Pertama buat S3Client.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

Buat Permintaan Buat Bucket.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```



```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class S3BucketOps {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String bucket = "bucket" + System.currentTimeMillis();
        System.out.println(bucket);
        createBucket(s3, bucket);
        performOperations(s3, bucket);
    }

    // Create a bucket by using a S3Waiter object
    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Daftar ember

Membangun a [ListBucketsRequest](#). Gunakan `listBuckets` metode `S3Client` untuk mengambil daftar bucket. Jika permintaan berhasil [ListBucketsResponse](#) dikembalikan. Gunakan objek respon ini untuk mengambil daftar ember.

### Impor

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

### Kode

Pertama buat `S3Client`.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

Buat Daftar Permintaan Bucket.

```
// List buckets
ListBucketsRequest listBucketsRequest = ListBucketsRequest.builder().build();
ListBucketsResponse listBucketsResponse = s3.listBuckets(listBucketsRequest);
listBucketsResponse.buckets().stream().forEach(x ->
System.out.println(x.name()));
```

Lihat [contoh lengkapnya](#) di GitHub.

## Hapus bucket

Sebelum Anda dapat menghapus Amazon S3 ember, Anda harus memastikan bahwa bucket kosong atau layanan akan mengembalikan kesalahan. Jika Anda memiliki [bucket berversi](#), Anda juga harus menghapus objek berversi apa pun yang ada di bucket.

### Topik

- [Hapus objek dalam ember](#)
- [Hapus ember kosong](#)

### Hapus objek dalam ember

Buat [ListObjectsV2Request](#) dan gunakan `listObjects` metode `S3Client` untuk mengambil daftar objek di bucket. Kemudian gunakan `deleteObject` metode pada setiap objek untuk menghapusnya.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
```

### Kode

Pertama buat `S3Client`.

```
ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .credentialsProvider(credentialsProvider)
    .build();
```

Hapus semua objek di ember.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3BucketDeletion {
    public static void main(String[] args) throws Exception {
        final String usage = ""

            Usage:
                <bucket>

            Where:
                bucket - The bucket to delete (for example, bucket1).\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteObjectsInBucket(s3, bucket);
        s3.close();
    }
}
```

```
public static void deleteObjectsInBucket(S3Client s3, String bucket) {
    try {
        // To delete a bucket, all the objects in the bucket must be deleted first.
        ListObjectsV2Request listObjectsV2Request = ListObjectsV2Request.builder()
            .bucket(bucket)
            .build();
        ListObjectsV2Response listObjectsV2Response;

        do {
            listObjectsV2Response = s3.listObjectsV2(listObjectsV2Request);
            for (S3Object s3Object : listObjectsV2Response.contents()) {
                DeleteObjectRequest request = DeleteObjectRequest.builder()
                    .bucket(bucket)
                    .key(s3Object.key())
                    .build();
                s3.deleteObject(request);
            }
        } while (listObjectsV2Response.isTruncated());
        DeleteBucketRequest deleteBucketRequest =
        DeleteBucketRequest.builder().bucket(bucket).build();
        s3.deleteBucket(deleteBucketRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

Hapus ember kosong

Buat [DeleteBucketRequest](#) dengan nama bucket dan teruskan ke metode S3Client. deleteBucket

Impor

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
```

```
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

## Kode

Pertama buat S3Client.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

Hapus bucket.

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();

s3.deleteBucket(deleteBucketRequest);
s3.close();
```

Lihat [contoh lengkapnya](#) di GitHub.

## Bekerja dengan Amazon S3 benda

Amazon S3Objek mewakili file atau kumpulan data. Setiap benda harus terkandung dalam [ember](#).

### Note

#### Praktik Terbaik

Kami menyarankan Anda mengaktifkan aturan [AbortIncompleteMultipartUpload](#) siklus hidup pada bucket Anda Amazon S3.

Aturan ini mengarahkan Amazon S3 untuk membatalkan unggahan multipart yang tidak selesai dalam jumlah hari tertentu setelah dimulai. Ketika batas waktu yang ditetapkan terlampaui, Amazon S3 batalkan unggahan dan kemudian menghapus data unggahan yang tidak lengkap.

Untuk informasi selengkapnya, lihat [Konfigurasi Siklus Hidup untuk Bucket dengan Pembuatan Versi di Panduan Pengguna](#). Amazon Simple Storage Service

### Note

Cuplikan kode ini mengasumsikan bahwa Anda memahami materi dalam dasar-dasar, dan telah mengonfigurasi AWS kredensi default menggunakan informasi di [the section called “Menyiapkan akses masuk tunggal untuk SDK”](#)

## Topik

- [Mengunggah objek](#)
- [Unggah objek di beberapa bagian](#)
- [Menghapus objek](#)
- [Mencantumkan Objek](#)
- [Lebih banyak contoh](#)

## Mengunggah objek

Buat [PutObjectRequest](#) dan berikan nama bucket dan nama kunci. Kemudian gunakan `putObject` metode `S3Client` dengan [RequestBody](#) yang berisi konten objek dan objek. `PutObjectRequest` Bucket harus ada, atau layanan akan mengembalikan kesalahan.

## Impor

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
```

```
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

## Kode

```
Region region = Region.US_WEST_2;
s3 = S3Client.builder()
    .region(region)
    .build();

createBucket(s3, bucketName, region);

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

s3.putObject(objectRequest,
    RequestBody.fromByteBuffer(getRandomByteBuffer(10_000)));
```

Lihat [contoh lengkapnya](#) di GitHub.

## Unggah objek di beberapa bagian

Gunakan `createMultipartUpload` metode `S3Client` untuk mendapatkan ID unggahan. Kemudian gunakan `uploadPart` metode untuk mengunggah setiap bagian. Terakhir, gunakan `completeMultipartUpload` metode `S3Client` untuk memberitahu Amazon S3 untuk menggabungkan semua bagian yang diunggah dan menyelesaikan operasi upload.

## Impor



```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

## Kode

```
        // First create a multipart upload and get the upload id
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = response.uploadId();
        System.out.println(uploadId);

        // Upload all the different parts of the object
```

```
UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .partNumber(1).build();

String etag1 = s3
    .uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
    .eTag();

CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
    .uploadId(uploadId)
    .partNumber(2).build();

String etag2 = s3
    .uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB)))
    .eTag();

CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

// Finally call completeMultipartUpload operation to tell S3 to merge
all

// uploaded
// parts and finish the multipart operation.
CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
    .parts(part1, part2)
    .build();

CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .multipartUpload(completedMultipartUpload)
    .build();

s3.completeMultipartUpload(completeMultipartUploadRequest);
```

Lihat [contoh lengkapnya](#) di GitHub.

## Menghapus objek

Buat [DeleteObjectRequest](#) dan berikan nama bucket dan nama kunci. Gunakan `deleteObject` metode `S3Client`, dan berikan nama bucket dan objek yang akan dihapus. Bucket dan kunci objek yang ditentukan harus ada, atau layanan akan mengembalikan kesalahan.

### Impor

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

### Kode

```
DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
    .bucket(bucketName)
    .key(key)
```

```
        .build();

s3.deleteObject(deleteObjectRequest);
```

Lihat [contoh lengkapnya](#) di GitHub.

## Menyalin objek

Buat [CopyObjectRequest](#) dan berikan nama bucket tempat objek diatasi, nilai string yang dikodekan URL (lihat metode `URLencoder.encode`), dan nama kunci objek. Gunakan `copyObject` metode `S3Client`, dan lewati objek. [CopyObjectRequest](#) Bucket dan kunci objek yang ditentukan harus ada, atau layanan akan mengembalikan kesalahan.

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

## Kode

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CopyObject {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
```

```
<objectKey> <fromBucket> <toBucket>
```

Where:

objectKey - The name of the object (for example, book.pdf).

fromBucket - The S3 bucket name that contains the object (for example, bucket1).

toBucket - The S3 bucket to copy the object to (for example, bucket2).

```
""";
```

```
if (args.length != 3) {
    System.out.println(usage);
    System.exit(1);
}
```

```
String objectKey = args[0];
String fromBucket = args[1];
String toBucket = args[2];
System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

copyBucketObject(s3, fromBucket, objectKey, toBucket);
s3.close();
}
```

```
public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .sourceBucket(fromBucket)
        .sourceKey(objectKey)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        return copyRes.copyObjectResult().toString();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
    return "";
}
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Mencantumkan Objek

Bangun [ListObjectsRequest](#) dan berikan nama bucket. Kemudian panggil `listObjects` metode `S3Client` dan lewati objek. `ListObjectsRequest` Metode ini mengembalikan a [ListObjectsResponse](#) yang berisi semua objek dalam ember. Anda dapat memanggil metode isi objek ini untuk mendapatkan daftar objek. Anda dapat mengulangi melalui daftar ini untuk menampilkan objek, seperti yang ditunjukkan dalam contoh kode berikut.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;
```

### Kode

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class ListObjects {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                .builder()
                .bucket(bucketName)
                .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.print("\n The name of the key is " + myValue.key());
                System.out.print("\n The object is " + calKb(myValue.size()) + " KBs");
                System.out.print("\n The owner is " + myValue.owner());
            }
        }
    }
}
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // convert bytes to kbs.
    private static long calKb(Long val) {
        return val / 1024;
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Lebih banyak contoh

Bagian [contoh Kode](#) dari panduan ini berisi lebih banyak contoh bekerja dengan objek Amazon S3 termasuk cara [mengunduh objek](#).

## Bekerja dengan URL yang Amazon S3 telah ditandatangani sebelumnya

URL yang telah ditandatangani sebelumnya menyediakan akses sementara ke objek S3 pribadi tanpa mengharuskan pengguna memiliki AWS kredensial atau izin.

Misalnya, asumsikan Alice memiliki akses ke objek S3, dan dia ingin sementara berbagi akses ke objek itu dengan Bob. Alice dapat membuat permintaan GET yang telah ditandatangani sebelumnya untuk dibagikan dengan Bob sehingga ia dapat mengunduh objek tanpa memerlukan akses ke kredensial Alice. Anda dapat membuat URL pra-ditandatangani untuk HTTP GET dan untuk permintaan HTTP PUT.

Buat URL yang telah ditandatangani sebelumnya untuk suatu objek, lalu unduh (GET request)

Contoh berikut terdiri dari dua bagian.

- Bagian 1: Alice menghasilkan URL pra-ditandatangani untuk objek.
- Bagian 2: Bob mengunduh objek dengan menggunakan URL yang telah ditandatangani sebelumnya.



## Bagian 1: Menghasilkan URL

Alice sudah memiliki objek di ember S3. Dia menggunakan kode berikut untuk menghasilkan string URL yang dapat digunakan Bob dalam permintaan GET berikutnya.

### Impor

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

```
/* Create a pre-signed URL to download an object in a subsequent GET request. */
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
```

```

        .key(keyName)
        .build();

    GetObjectPresignRequest presignRequest = GetObjectPresignRequest.builder()
        .signatureDuration(Duration.ofMinutes(10)) // The URL will expire
in 10 minutes.
        .getObjectRequest(objectRequest)
        .build();

    PresignedGetObjectRequest presignedRequest =
presigner.presignGetObject(presignRequest);
    logger.info("Presigned URL: [{}]", presignedRequest.url().toString());
    logger.info("HTTP method: [{}]", presignedRequest.httpRequest().method());

    return presignedRequest.url().toExternalForm();
}
}

```

## Bagian 2: Unduh objek

Bob menggunakan salah satu dari tiga opsi kode berikut untuk mengunduh objek. Atau, dia bisa menggunakan browser untuk melakukan permintaan GET.

### Gunakan JDK **HttpURLConnection** (sejak v1.1)

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
Capture the response body to a byte array.

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setRequestMethod("GET");
        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

```
    }  
    return byteArrayOutputStream.toByteArray();  
}
```

## Gunakan JDK **HttpClient** (sejak v11)

```
/* Use the JDK HttpClient (since v11) class to do the download. */  
public byte[] useHttpClientToGet(String presignedUrlString) {  
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //  
    Capture the response body to a byte array.  
  
    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();  
    HttpClient httpClient = HttpClient.newHttpClient();  
    try {  
        URL presignedUrl = new URL(presignedUrlString);  
        HttpResponse<InputStream> response = httpClient.send(requestBuilder  
            .uri(presignedUrl.toURI())  
            .GET()  
            .build(),  
            HttpResponse.BodyHandlers.ofInputStream());  
  
        IoUtils.copy(response.body(), byteArrayOutputStream);  
  
        logger.info("HTTP response code is " + response.statusCode());  
    } catch (URISyntaxException | InterruptedException | IOException e) {  
        logger.error(e.getMessage(), e);  
    }  
    return byteArrayOutputStream.toByteArray();  
}
```

## Gunakan **SdkHttpClient** dari SDK for Java

```
/* Use the AWS SDK for Java SdkHttpClient class to do the download. */  
public byte[] useSdkHttpClientToPut(String presignedUrlString) {  
  
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //  
    Capture the response body to a byte array.  
    try {  
        URL presignedUrl = new URL(presignedUrlString);  
        SdkHttpRequest request = SdkHttpRequest.builder()  
            .method(SdkHttpMethod.GET)  
            .uri(presignedUrl.toURI())
```

```
        .build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
                sdkHttpClient.prepareRequest(executeRequest).call();
            response.responseBody().ifPresentOrElse(
                abortableInputStream -> {
                    try {
                        IoUtils.copy(abortableInputStream,
                            byteArrayOutputStream);
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                },
                () -> logger.error("No response body."));

            logger.info("HTTP Response code is {}",
                response.httpResponse().statusCode());
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
}
```

Lihat [contoh lengkapnya](#) dan [uji](#) GitHub.

Buat URL yang telah ditandatangani sebelumnya untuk unggahan, lalu unggah file (permintaan PUT)

Contoh berikut terdiri dari dua bagian.

- Bagian 1: Alice menghasilkan URL yang telah ditandatangani sebelumnya untuk mengunggah objek.
- Bagian 2: Bob mengunggah file dengan menggunakan URL yang telah ditandatangani sebelumnya.

## Bagian 1: Menghasilkan URL

Alice sudah memiliki ember S3. Dia menggunakan kode berikut untuk menghasilkan string URL yang dapat digunakan Bob dalam permintaan PUT berikutnya.

### Impor

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

```
/* Create a presigned URL to use in a subsequent PUT request */
```

```

public String createPresignedUrl(String bucketName, String keyName, Map<String,
String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .metadata(metadata)
            .build();

        PutObjectPresignRequest presignRequest = PutObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL expires in
10 minutes.
            .putObjectRequest(objectRequest)
            .build();

        PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload a file to: [{}]", myURL);
        logger.info("HTTP method: [{}]", presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

## Bagian 2: Unggah objek file

Bob menggunakan salah satu dari tiga opsi kode berikut untuk mengunggah file.

Gunakan JDK **HttpURLConnection** (sejak v1.1)

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
public void useHttpURLConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" + k,
v));
    }
}

```

```

connection.setRequestMethod("PUT");
OutputStream out = connection.getOutputStream();

try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
    FileChannel inChannel = file.getChannel()) {
    ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

    while (inChannel.read(buffer) > 0) {
        buffer.flip();
        for (int i = 0; i < buffer.limit(); i++) {
            out.write(buffer.get());
        }
        buffer.clear();
    }
} catch (IOException e) {
    logger.error(e.getMessage(), e);
}

out.close();
connection.getResponseCode();
logger.info("HTTP response code is " + connection.getResponseCode());

} catch (S3Exception | IOException e) {
    logger.error(e.getMessage(), e);
}
}

```

## Gunakan JDK **HttpClient** (sejak v11)

```

/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        final HttpResponse<Void> response = httpClient.send(requestBuilder
            .uri(new URL(presignedUrlString).toURI())
            .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI()))))

```

```

        .build(),
        HttpResponse.BodyHandlers.discarding());

    logger.info("HTTP response code is " + response.statusCode());
} catch (URISyntaxException | InterruptedException | IOException e) {
    logger.error(e.getMessage(), e);
}
}

```

## Gunakan **SdkHttpClient** dari SDK for Java

```

/* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());
        // Add headers
        metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" + k, v));
        // Finish building the request.
        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            logger.info("Response code: {}", response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}
}

```



Lihat [contoh lengkapnya](#) dan [uji](#) GitHub.

## Akses Lintas Wilayah untuk Amazon S3

Saat Anda bekerja dengan bucket Amazon Simple Storage Service (Amazon S3), Anda biasanya tahu Wilayah AWS bucket tersebut. Wilayah tempat Anda bekerja ditentukan saat Anda membuat klien S3.

Namun, terkadang Anda mungkin perlu bekerja dengan bucket tertentu, tetapi Anda tidak tahu apakah itu terletak di Wilayah yang sama yang ditetapkan untuk klien S3.

Alih-alih melakukan lebih banyak panggilan untuk menentukan Wilayah bucket, Anda dapat menggunakan SDK untuk mengaktifkan akses ke bucket S3 di berbagai Wilayah.

### Penyiapan

Support untuk akses lintas wilayah tersedia dengan 2.20.111 versi SDK. Gunakan versi ini atau yang lebih baru di file build Maven Anda untuk s3 ketergantungan seperti yang ditunjukkan pada cuplikan berikut.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.20.111</version>
</dependency>
```

Selanjutnya, saat Anda membuat klien S3, aktifkan akses lintas wilayah seperti yang ditunjukkan pada cuplikan. Secara default, akses tidak diaktifkan.

```
S3AsyncClient client = S3AsyncClient.builder()
    .crossRegionAccessEnabled(true)
    .build();
```

### Bagaimana SDK menyediakan akses Lintas wilayah

Saat Anda mereferensikan bucket yang ada dalam permintaan, seperti saat Anda menggunakan `putObject` metode, SDK akan memulai permintaan ke Wilayah yang dikonfigurasi untuk klien.

Jika bucket tidak ada di Region tertentu, respons kesalahan menyertakan Region sebenarnya tempat bucket berada. SDK kemudian menggunakan Region yang benar dalam permintaan kedua.

Untuk mengoptimalkan permintaan future ke bucket yang sama, SDK menyimpan pemetaan Region ini di klien.

## Pertimbangan-pertimbangan

Saat Anda mengaktifkan akses bucket lintas wilayah, ketahuilah bahwa panggilan API pertama dapat meningkatkan latensi jika bucket tidak berada di Region yang dikonfigurasi klien. Namun, panggilan berikutnya mendapat manfaat dari informasi Wilayah yang di-cache, menghasilkan peningkatan kinerja.

Saat Anda mengaktifkan akses lintas wilayah, akses ke bucket tidak terpengaruh. Pengguna harus diberi wewenang untuk mengakses bucket di Wilayah mana pun tempat tinggalnya.

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](#).

Amazon S3 saat ini mendukung empat algoritma checksum: SHA-1, SHA-256, CRC-32, dan CRC-32C. Anda memiliki fleksibilitas untuk memilih algoritma yang paling sesuai dengan kebutuhan Anda dan membiarkan SDK menghitung checksum. Atau, Anda dapat menentukan nilai checksum yang telah dihitung sebelumnya dengan menggunakan salah satu dari empat algoritme yang didukung.

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

## Mengunggah objek

Nilai yang valid untuk algoritma adalah CRC32, CRC32C, SHA1, dan SHA256.

Cuplikan kode berikut menunjukkan permintaan untuk mengunggah objek dengan checksum CRC-32. Ketika SDK mengirimkan permintaan, ia menghitung checksum CRC-32 dan mengunggah objek. Amazon S3 menyimpan checksum dengan objek.

Jika checksum yang dihitung SDK tidak cocok dengan checksum yang dihitung Amazon S3 saat menerima permintaan, kesalahan akan dikembalikan.

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 checksum SHA-256 yang telah dihitung sebelumnya.

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.

Unduh objek

Saat Anda menggunakan metode [getObject](#) untuk mengunduh objek, SDK secara otomatis memvalidasi checksum `.enabled`

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

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

Objek di Amazon S3 dapat memiliki beberapa checksum, tetapi hanya satu checksum yang divalidasi saat diunduh. Prioritas berikut—berdasarkan efisiensi algoritma checksum—menentukan checksum mana yang divalidasi SDK:

1. CRC-32C
2. CRC-32
3. SHA-1
4. SHA-256

Misalnya, jika respons berisi checksum CRC-32 dan SHA-256, hanya checksum CRC-32 yang divalidasi.

**Gunakan klien S3 berkinerja: klien S3 berbasis AWS CRT**

Klien S3 AWS berbasis CRT — dibangun di atas [AWSCommon Runtime \(CRT\)](#) — adalah klien [asinkron](#) S3 alternatif. [Ini mentransfer objek ke dan dari Amazon Simple Storage Service \(Amazon](#)

### [S3\) Simple Storage Service \(Amazon S3\) dengan peningkatan kinerja dan keandalan dengan secara otomatis menggunakan API unggahan multipart Amazon S3 dan pengambilan rentang byte.](#)

Klien S3 AWS berbasis CRT meningkatkan keandalan transfer jika ada kegagalan jaringan. Keandalan ditingkatkan dengan mencoba kembali bagian-bagian individual yang gagal dari transfer file tanpa memulai ulang transfer dari awal.

Selain itu, klien S3 AWS berbasis CRT menawarkan penyatuan koneksi yang ditingkatkan dan penyeimbangan beban Sistem Nama Domain (DNS), yang juga meningkatkan throughput.

Anda dapat menggunakan klien S3 AWS berbasis CRT sebagai pengganti klien asinkron S3 standar SDK dan segera memanfaatkan throughputnya yang ditingkatkan.

AWSKomponen berbasis CRT di SDK

Klien S3 AWS berbasis CRT, dijelaskan dalam topik ini, dan klien HTTP AWS berbasis CRT adalah komponen yang berbeda dalam SDK.

Klien S3 AWS berbasis CRT adalah implementasi `AsyncClient` antarmuka [S3](#) dan digunakan untuk bekerja dengan layanan Amazon S3. Ini adalah alternatif untuk implementasi `S3AsyncClient` antarmuka berbasis Java dan menawarkan beberapa manfaat.

[Client HTTP AWS berbasis CRT](#) adalah implementasi `SdkAsyncHttpClient` antarmuka dan digunakan untuk komunikasi HTTP umum. Ini adalah alternatif untuk implementasi `SdkAsyncHttpClient` antarmuka Netty dan menawarkan beberapa keuntungan.

Meskipun kedua komponen menggunakan pustaka dari [AWSCommon Runtime](#), klien S3 AWS berbasis CRT menggunakan [pustaka aws-c-s 3](#) dan mendukung fitur API unggahan multipart [S3](#). Karena klien HTTP AWS berbasis CRT dimaksudkan untuk penggunaan tujuan umum, itu tidak mendukung fitur API unggahan multibagian S3.

### Tambahkan dependensi untuk menggunakan klien S3 berbasis AWS CRT

Untuk menggunakan klien S3 AWS berbasis CRT, tambahkan dua dependensi berikut ke file proyek Maven Anda. Contoh menunjukkan versi minimum yang akan digunakan. [Cari repositori pusat Maven untuk versi terbaru dari artefak s3 dan aws-crt.](#)

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.20.68</version>
</dependency>
```

```
<dependency>
  <groupId>software.amazon.awssdk.crt</groupId>
  <artifactId>aws-crt</artifactId>
  <version>0.21.16</version>
</dependency>
```

## Buat instance klien S3 AWS berbasis CRT

Buat instance klien S3 AWS berbasis CRT dengan pengaturan default seperti yang ditunjukkan pada cuplikan kode berikut.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
```

Untuk mengkonfigurasi klien, gunakan pembuat klien AWS CRT. Anda dapat beralih dari klien asinkron S3 standar ke klien AWS berbasis CRT dengan mengubah metode pembangun.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;

S3AsyncClient s3AsyncClient =
    S3AsyncClient.crtBuilder()
        .credentialsProvider(DefaultCredentialsProvider.create())
        .region(Region.US_WEST_2)
        .targetThroughputInGbps(20.0)
        .minimumPartSizeInBytes(8 * 1025 * 1024L)
        .build();
```

### Note

Beberapa pengaturan di pembuat standar mungkin saat ini tidak didukung di pembuat klien AWS CRT. Dapatkan pembangun standar dengan `meneleponS3AsyncClient#builder()`.

## Gunakan klien AWS S3 berbasis CRT

Gunakan klien S3 AWS berbasis CRT untuk memanggil operasi API Amazon S3. Contoh berikut menunjukkan [PutObject](#) dan [GetObject](#) operasi yang tersedia melalui AWS SDK for Java

```
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

S3AsyncClient s3Client = S3AsyncClient.crtCreate();

// Upload a local file to Amazon S3.
PutObjectResponse putObjectResponse =
    s3Client.putObject(req -> req.bucket(<BUCKET_NAME>)
        .key(<KEY_NAME>),
        AsyncRequestBody.fromFile(Paths.get(<FILE_NAME>)))
        .join();

// Download an object from Amazon S3 to a local file.
GetObjectResponse getObjectResponse =
    s3Client.getObject(req -> req.bucket(<BUCKET_NAME>)
        .key(<KEY_NAME>),
        AsyncResponseTransformerToFile(Paths.get(<FILE_NAME>)))
        .join();
```

## Transfer file dan direktori dengan Amazon S3 Transfer Manager

Amazon S3 Transfer Manager adalah sumber terbuka, utilitas transfer file tingkat tinggi untuk file. AWS SDK for Java 2.x Gunakan untuk mentransfer file dan direktori ke dan dari Amazon Simple Storage Service (Amazon S3).

[Ketika dibangun di atas klien S3 AWS berbasis CRT, S3 Transfer Manager dapat memanfaatkan peningkatan kinerja seperti API unggahan multibagian dan pengambilan rentang byte.](#)

Dengan S3 Transfer Manager, Anda juga dapat memantau kemajuan transfer secara real time dan menunda transfer untuk eksekusi nanti.

### Memulai

Tambahkan dependensi ke file build Anda

Untuk menggunakan S3 Transfer Manager dengan peningkatan kinerja berdasarkan klien S3 berbasis AWS CRT, konfigurasi file build Anda dengan dependensi berikut.

- Gunakan versi **2.19.1** atau lebih tinggi dari SDK for Java 2.x.
- Tambahkan `s3-transfer-manager` artefak sebagai dependensi.
- Tambahkan `aws-crt` artefak sebagai dependensi pada versi **0.20.3** atau lebih tinggi.

Contoh kode berikut menunjukkan cara mengkonfigurasi dependensi proyek Anda untuk Maven.

```
<project>
  <properties>
    <aws.sdk.version>2.19.1</aws.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>s3-transfer-manager</artifactId>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk.crt</groupId>
      <artifactId>aws-crt</artifactId>
      <version>0.20.3</version>
    </dependency>
  </dependencies>
</project>
```

[Cari repositori pusat Maven untuk versi terbaru dari artefak `s3-transfer-manager` dan `aws-crt`.](#)

### Buat instance dari S3 Transfer Manager

Cuplikan berikut menunjukkan cara membuat `TransferManager` instance [S3](#) dengan pengaturan default.

```
S3TransferManager transferManager = S3TransferManager.create();
```

Contoh berikut menunjukkan cara mengkonfigurasi Manajer Transfer S3 dengan pengaturan khusus. Dalam contoh ini, AsyncClient instance [S3 AWS berbasis CRT](#) digunakan sebagai klien yang mendasari S3 Transfer Manager.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .targetThroughputInGbps(20.0)
    .minimumPartSizeInBytes(8 * MB)
    .build();

S3TransferManager transferManager = S3TransferManager.builder()
    .s3Client(s3AsyncClient)
    .build();
```

#### Note

Jika `aws-crt` dependensi tidak disertakan dalam file build, S3 Transfer Manager dibangun di atas klien asinkron S3 standar yang digunakan dalam SDK for Java 2.x.

## Unggah file ke bucket S3

Contoh berikut menunjukkan contoh upload file bersama dengan penggunaan opsional dari [LoggingTransferListener](#), yang mencatat kemajuan upload.

[Untuk mengunggah file ke Amazon S3 menggunakan S3 Transfer Manager, teruskan UploadFileRequest objek ke metode UploadFile. S3TransferManager](#)

[FileUpload](#) Objek yang dikembalikan dari `uploadFile` metode mewakili proses upload. Setelah permintaan selesai, [CompletedFileUpload](#) objek berisi informasi tentang unggahan.

```
public String uploadFile(S3TransferManager transferManager, String bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create())
        .source(Paths.get(filePathURI))
```



```

        .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }

```

## Impor

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

```

## Unduh file dari ember S3

Contoh berikut menunjukkan contoh download bersama dengan penggunaan opsional dari [LoggingTransferListener](#), yang mencatat kemajuan download.

Untuk mengunduh objek dari bucket S3 menggunakan S3 Transfer Manager, buat [DownloadFileRequest](#) objek dan teruskan ke metode [DownloadFile](#).

[FileDownload](#) Objek yang `S3TransferManager` dikembalikan oleh `downloadFile` metode ini mewakili transfer file. Setelah unduhan selesai, [CompletedFileDownload](#) berisi akses ke informasi tentang unduhan.

```

public Long downloadFile(S3TransferManager transferManager, String bucketName,
                        String key, String downloadedFileWithPath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create())
        .destination(Paths.get(downloadedFileWithPath))
        .build();

```

```

FileDownload downloadFile = transferManager.downloadFile(downloadFileRequest);

CompletedFileDownload downloadResult = downloadFile.completionFuture().join();
logger.info("Content length [{}]", downloadResult.response().contentLength());
return downloadResult.response().contentLength();
}

```

## Impor

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

```

## Salin objek Amazon S3 ke ember lain

Contoh berikut menunjukkan cara menyalin objek dengan S3 Transfer Manager.

Untuk memulai salinan objek dari bucket S3 ke bucket lain, buat [CopyObjectRequest](#) instance dasar.

Selanjutnya, bungkus dasar CopyObjectRequest dalam [CopyRequest](#) yang dapat digunakan oleh S3 Transfer Manager.

CopyObjek yang dikembalikan oleh copy metode ini mewakili proses penyalinan.

S3TransferManager Setelah proses penyalinan selesai, [CompletedCopy](#) objek berisi detail tentang respons.

```

public String copyObject(S3TransferManager transferManager, String bucketName,
    String key, String destinationBucket, String destinationKey) {
    CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
        .sourceBucket(bucketName)
        .sourceKey(key)

```

```
        .destinationBucket(destinationBucket)
        .destinationKey(destinationKey)
        .build();

CopyRequest copyRequest = CopyRequest.builder()
    .copyObjectRequest(copyObjectRequest)
    .build();

Copy copy = transferManager.copy(copyRequest);

CompletedCopy completedCopy = copy.completionFuture().join();
return completedCopy.response().copyObjectResult().eTag();
}
```

### Note

Untuk melakukan salinan Lintas wilayah dengan S3 Transfer Manager, aktifkan `crossRegionAccessEnabled` pada pembuat klien S3 AWS berbasis CRT seperti yang ditunjukkan pada cuplikan berikut.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
    .crossRegionAccessEnabled(true)
    .build();

S3TransferManager transferManager = S3TransferManager.builder()
    .s3Client(s3AsyncClient)
    .build();
```

## Impor

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;
```

## Unggah direktori lokal ke bucket S3

Contoh berikut menunjukkan bagaimana Anda dapat meng-upload direktori lokal ke S3.

Mulailah dengan memanggil metode [uploadDirectory](#) dari `S3TransferManager` instance, meneruskan file. [UploadDirectoryRequest](#)

[DirectoryUpload](#) objek mewakili proses upload, yang menghasilkan [CompletedDirectoryUpload](#) ketika permintaan selesai. `CompletedDirectoryUpload` objek berisi informasi tentang hasil transfer, termasuk file mana yang gagal ditransfer.

```
public Integer uploadDirectory(S3TransferManager transferManager,
    URI sourceDirectory, String bucketName) {
    DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
    .source(Paths.get(sourceDirectory))
    .bucket(bucketName)
    .build());

    CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
    completedDirectoryUpload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryUpload.failedTransfers().size();
}
```

### Impor

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;
```

## Unduh objek bucket S3 ke direktori lokal

Anda dapat mengunduh objek dalam bucket S3 ke direktori lokal seperti yang ditunjukkan pada contoh berikut.

Untuk mengunduh objek dalam bucket S3 ke direktori lokal, mulailah dengan memanggil metode [DownloadDirectory](#) dari Transfer Manager, meneruskan file. [DownloadDirectoryRequest](#)

[DirectoryDownload](#) objek mewakili proses download, yang menghasilkan [CompletedDirectoryDownload](#) ketika permintaan selesai. `CompletedDirectoryDownload` objek berisi informasi tentang hasil transfer, termasuk file mana yang gagal ditransfer.

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
    .destination(Paths.get(destinationPathURI))
    .bucket(bucketName)
    .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}
```

### Impor

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;

import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
```

```
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;
```

## Lihat contoh lengkap

[GitHub berisi kode lengkap](#) untuk semua contoh di halaman ini.

# Bekerja dengan Amazon Simple Notification Service

Dengan Amazon Simple Notification Service, Anda dapat dengan mudah mendorong pesan notifikasi real-time dari aplikasi Anda ke pelanggan melalui beberapa saluran komunikasi. Topik ini menjelaskan bagaimana melakukan beberapa fungsi dasar Amazon SNS.

## Buat topik

Topik adalah pengelompokan logis saluran komunikasi yang mendefinisikan sistem mana yang akan mengirim pesan, misalnya, mengipasi pesan AWS Lambda dan webhook HTTP. Anda mengirim pesan ke Amazon SNS, lalu mereka didistribusikan ke saluran yang ditentukan dalam topik. Ini membuat pesan tersedia untuk pelanggan.

Untuk membuat topik, pertama-tama buat [CreateTopicRequest](#) objek, dengan nama set topik menggunakan `name()` metode di pembuat. Kemudian, kirim objek permintaan ke Amazon SNS dengan menggunakan `createTopic()` metode [SnsClient](#). Anda dapat menangkap hasil permintaan ini sebagai [CreateTopicResponse](#) objek, seperti yang ditunjukkan dalam cuplikan kode berikut.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

### Kode

```
public static String createSNSTopic(SnsClient snsClient, String topicName ) {
```

```
    CreateTopicResponse result = null;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();
    } catch (SnsException e) {

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Buat daftar Amazon SNS topik Anda

Untuk mengambil daftar Amazon SNS topik yang ada, buat [ListTopicsRequest](#) objek. Kemudian, kirim objek permintaan ke Amazon SNS dengan menggunakan `listTopics()` metode `SnsClient`. Anda dapat menangkap hasil permintaan ini sebagai [ListTopicsResponse](#) objek.

Cuplikan kode berikut mencetak kode status HTTP permintaan dan daftar Nama Sumber Daya Amazon (ARN) untuk topik Anda. Amazon SNS

Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Kode

```
public static void listSNSTopics(SnsClient snsClient) {

    try {
        ListTopicsRequest request = ListTopicsRequest.builder()
```

```
        .build();

        ListTopicsResponse result = snsClient.listTopics(request);
        System.out.println("Status was " + result.sdkHttpResponse().statusCode() +
"\n\nTopics\n\n" + result.topics());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Berlangganan titik akhir ke suatu topik

Setelah Anda membuat topik, Anda dapat mengonfigurasi saluran komunikasi mana yang akan menjadi titik akhir untuk topik tersebut. Pesan didistribusikan ke titik akhir ini setelah Amazon SNS menerimanya.

Untuk mengonfigurasi saluran komunikasi sebagai titik akhir untuk suatu topik, berlangganan titik akhir tersebut ke topik tersebut. Untuk memulai, buat [SubscribeRequest](#) objek. Tentukan saluran komunikasi (misalnya, lambda atau email) sebagai `protocol()`. Atur `endpoint()` ke lokasi keluaran yang relevan (misalnya, ARN Lambda fungsi atau alamat email), lalu atur ARN dari topik yang ingin Anda berlangganan sebagai `topicArn()`. Kirim objek permintaan ke Amazon SNS dengan menggunakan `subscribe()` metode `SnsClient`. Anda dapat menangkap hasil permintaan ini sebagai [SubscribeResponse](#) objek.

Cuplikan kode berikut menunjukkan cara berlangganan alamat email ke suatu topik.

Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
```

Kode

```
public static void subEmail(SnsClient snsClient, String topicArn, String email) {
```



```
try {
    SubscribeRequest request = SubscribeRequest.builder()
        .protocol("email")
        .endpoint(email)
        .returnSubscriptionArn(true)
        .topicArn(topicArn)
        .build();

    SubscribeResponse result = snsClient.subscribe(request);
    System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n\n"
        + "Status is " + result.sdkHttpResponse().statusCode());

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Publikasikan pesan ke suatu topik

Setelah Anda memiliki topik dan satu atau beberapa titik akhir yang dikonfigurasi untuk itu, Anda dapat mempublikasikan pesan ke sana. Untuk memulai, buat [PublishRequest](#) objek. Tentukan `message()` yang akan dikirim, dan ARN dari topik (`topicArn()`) untuk mengirimkannya. Kemudian, kirim objek permintaan ke Amazon SNS dengan menggunakan `publish()` metode `SnsClient`. Anda dapat menangkap hasil permintaan ini sebagai [PublishResponse](#) objek.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

### Kode

```
public static void pubTopic(SnsClient snsClient, String message, String topicArn) {

    try {
```

```
    PublishRequest request = PublishRequest.builder()
        .message(message)
        .topicArn(topicArn)
        .build();

    PublishResponse result = snsClient.publish(request);
    System.out.println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Berhenti berlangganan titik akhir dari suatu topik

Anda dapat menghapus saluran komunikasi yang dikonfigurasi sebagai titik akhir untuk suatu topik. Setelah melakukan itu, topik itu sendiri terus ada dan mendistribusikan pesan ke titik akhir lain yang dikonfigurasi untuk topik itu.

Untuk menghapus saluran komunikasi sebagai titik akhir topik, berhenti berlangganan titik akhir dari topik tersebut. Untuk memulai, buat [UnsubscribeRequest](#) objek dan atur ARN dari topik yang ingin Anda berhenti berlangganan sebagai `subscriptionArn()` Kemudian kirim objek permintaan ke SNS dengan menggunakan `unsubscribe()` metode. `SnsClient` Anda dapat menangkap hasil permintaan ini sebagai [UnsubscribeResponse](#) objek.

Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
```

Kode

```
public static void unSub(SnsClient snsClient, String subscriptionArn) {

    try {
```

```
UnsubscribeRequest request = UnsubscribeRequest.builder()
    .subscriptionArn(subscriptionArn)
    .build();

UnsubscribeResponse result = snsClient.unsubscribe(request);

System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode()
    + "\n\nSubscription was removed for " + request.subscriptionArn());

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Hapus topik

Untuk menghapus Amazon SNS topik, pertama-tama buat [DeleteTopicRequest](#) objek dengan ARN dari set topik sebagai `topicArn()` metode di pembuat. Kemudian kirim objek permintaan ke Amazon SNS dengan menggunakan `deleteTopic()` metode `SnsClient`. Anda dapat menangkap hasil permintaan ini sebagai [DeleteTopicResponse](#) objek, seperti yang ditunjukkan dalam cuplikan kode berikut.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

### Kode

```
public static void deleteSNSTopic(SnsClient snsClient, String topicArn ) {

    try {
        DeleteTopicRequest request = DeleteTopicRequest.builder()
            .topicArn(topicArn)
            .build();
```

```
        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

Untuk informasi selengkapnya, lihat [Panduan Developer Amazon Simple Notification Service](#).

## Bekerja dengan Amazon Simple Queue Service

Bagian ini memberikan contoh pemrograman [Amazon Simple Queue Service](#) menggunakan AWS SDK for Java 2.x.

Contoh berikut hanya mencakup kode yang diperlukan untuk mendemonstrasikan setiap teknik. [Kode contoh lengkap tersedia di GitHub](#). Dari sana, Anda dapat mengunduh satu file sumber atau mengkloning repositori secara lokal untuk mendapatkan semua contoh untuk dibangun dan dijalankan.

Topik

- [Bekerja dengan Amazon Simple Queue Service antrian pesan](#)
- [Mengirim, menerima, dan menghapus Amazon Simple Queue Service pesan](#)

### Bekerja dengan Amazon Simple Queue Service antrian pesan

Antrian pesan adalah wadah logis yang digunakan untuk mengirim pesan dengan andal. Amazon Simple Queue Service Ada dua jenis antrian: standar dan first-in, first-out (FIFO). Untuk mempelajari lebih lanjut tentang antrian dan perbedaan di antara jenis-jenis ini, lihat Panduan [Amazon Simple Queue Service Pengembang](#).

Topik ini menjelaskan cara membuat, membuat daftar, menghapus, dan mendapatkan URL Amazon Simple Queue Service antrian dengan menggunakan. AWS SDK for Java

`sqsClient` Variabel yang digunakan dalam contoh berikut dapat dibuat dari cuplikan berikut.

```
SqsClient sqsClient = SqsClient.create();
```

Bila Anda membuat `SqsClient` dengan menggunakan `create()` metode statis, SDK akan mengonfigurasi Region menggunakan [rantai penyedia wilayah default dan kredensialnya menggunakan rantai penyedia](#) kredensial [default](#).

## Membuat antrian

Gunakan `SqsClient`'s `createQueue` metode ini, dan berikan [CreateQueueRequest](#) objek yang menjelaskan parameter antrian seperti yang ditunjukkan pada cuplikan kode berikut.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

### Kode

```
CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
    .queueName(queueName)
    .build();

sqsClient.createQueue(createQueueRequest);
```

Lihat [sampel lengkapnya](#) di GitHub.

## Daftar antrian

Untuk membuat daftar Amazon Simple Queue Service antrian untuk akun Anda, panggil `SqsClient`'s `listQueues` metode dengan [ListQueuesRequest](#) objek.

Bila Anda menggunakan bentuk [listQueues](#) metode yang tidak mengambil parameter, layanan mengembalikan semua antrian hingga 1.000 antrian.

Anda dapat memberikan awalan nama antrian ke [ListQueuesRequest](#) objek untuk membatasi hasil ke antrian yang cocok dengan awalan seperti yang ditunjukkan dalam kode berikut.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Kode

```
String prefix = "que";

try {
    ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
    ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);

    for (String url : listQueuesResponse.queueUrls()) {
        System.out.println(url);
    }

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

Lihat [sampel lengkapnya](#) di GitHub.

## Dapatkan URL untuk antrian

Kode berikut menunjukkan cara mendapatkan URL untuk antrian dengan memanggil SqsClient's `getQueueUrl` metode dengan [GetQueueUrlRequest](#) objek.

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Kode

```
GetQueueUrlResponse getQueueUrlResponse =
```

```
sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
String queueUrl = getQueueUrlResponse.queueUrl();
return queueUrl;
```

Lihat [sampel lengkapnya](#) di GitHub.

## Hapus antrian

Berikan [URL](#) antrian ke [DeleteQueueRequest](#) objek. Kemudian panggil SqsClient's `deleteQueue` metode untuk menghapus antrian seperti yang ditunjukkan pada kode berikut.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

### Kode

```
public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {

    try {

        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();

        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [sampel lengkapnya](#) di GitHub.

## Informasi lain

- [CreateQueue](#) di Referensi Amazon Simple Queue Service API
- [GetQueueUrl](#) di Referensi Amazon Simple Queue Service API
- [ListQueues](#) di Referensi Amazon Simple Queue Service API
- [DeleteQueue](#) di Referensi Amazon Simple Queue Service API

## Mengirim, menerima, dan menghapus Amazon Simple Queue Service pesan

Pesan adalah sepotong data yang dapat dikirim dan diterima oleh komponen terdistribusi. Pesan selalu dikirimkan menggunakan [SQS Queue](#).

`sqsClient` variabel yang digunakan dalam contoh berikut dapat dibuat dari cuplikan berikut.

```
SqsClient sqsClient = SqsClient.create();
```

Bila Anda membuat `SqsClient` dengan menggunakan `create()` metode statis, SDK akan mengonfigurasi Region menggunakan [rantai penyedia wilayah default dan kredensialnya menggunakan rantai penyedia](#) kredensial [default](#).

## Kirim pesan

Tambahkan satu pesan ke Amazon Simple Queue Service antrian dengan memanggil `sendMessage` metode `SqsClient` klien. Berikan [SendMessageRequest](#) objek yang berisi [URL](#) antrian, isi pesan, dan nilai penundaan opsional (dalam hitungan detik).

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Kode

```
sqsClient.sendMessage(SendMessageRequest.builder()
```



```
        .queueUrl(queueUrl)
        .messageBody("Hello world!")
        .delaySeconds(10)
        .build());

sqsClient.sendMessage(sendMsgRequest);
```

## Kirim beberapa pesan dalam satu permintaan

Kirim lebih dari satu pesan dalam satu permintaan dengan menggunakan `SqsClient` `sendMessageBatch` metode ini. Metode ini mengambil [SendMessageBatchRequest](#) yang berisi URL antrian dan daftar pesan untuk dikirim. (Setiap pesan adalah a [SendMessageBatchRequestEntry](#).) Anda juga dapat menunda pengiriman pesan tertentu dengan menetapkan nilai penundaan pada pesan.

### Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

### Kode

```
SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
    .queueUrl(queueUrl)

    .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from msg
1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10).build())
    .build();
sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

Lihat [sampel lengkapnya](#) di GitHub.

## Ambil Pesan

Ambil pesan apa pun yang saat ini berada dalam antrian dengan memanggil metode `SqsClient` `receiveMessage` Metode ini mengambil [ReceiveMessageRequest](#) yang berisi URL antrian. Anda

juga dapat menentukan jumlah maksimum pesan yang akan dikembalikan. Pesan dikembalikan sebagai daftar objek [Pesan](#).

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Kode

```
try {
    ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .maxNumberOfMessages(5)
        .build();
    List<Message> messages =
sqsClient.receiveMessage(receiveMessageRequest).messages();
    return messages;
} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
```

Lihat [sampel lengkapnya](#) di GitHub.

## Hapus pesan setelah diterima

Setelah menerima pesan dan memproses isinya, hapus pesan dari antrian dengan mengirimkan alamat tanda terima pesan dan URL antrian ke metode. `SqsClient` 's [deleteMessage](#)

## Impor

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

## Kode

```
try {
    for (Message message : messages) {
        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
        .queueUrl(queueUrl)
        .receiptHandle(message.receiptHandle())
        .build();
        sqsClient.deleteMessage(deleteMessageRequest);
    }
}
```

Lihat [sampel lengkapnya](#) di GitHub.

## Info Selengkapnya

- [Cara Kerja Amazon Simple Queue Service Antrian di Panduan](#) Pengembang Amazon Simple Queue Service
- [SendMessage](#) di Referensi Amazon Simple Queue Service API
- [SendMessageBatch](#) di Referensi Amazon Simple Queue Service API
- [ReceiveMessage](#) di Referensi Amazon Simple Queue Service API
- [DeleteMessage](#) di Referensi Amazon Simple Queue Service API

## Bekerja dengan Amazon Transcribe

Contoh berikut menunjukkan cara kerja streaming dua arah menggunakan Amazon Transcribe Streaming dua arah berarti bahwa ada aliran data yang masuk ke layanan dan diterima kembali secara real time. Contoh menggunakan transkripsi Amazon Transcribe streaming untuk mengirim aliran audio dan menerima aliran teks yang ditranskripsi kembali secara real time.

Lihat [Transkripsi Streaming](#) di Panduan Amazon Transcribe Pengembang untuk mempelajari lebih lanjut tentang fitur ini.

Lihat [Memulai](#) di Panduan Amazon Transcribe Pengembang untuk mulai menggunakan Amazon Transcribe.

## Siapkan mikrofon

Kode ini menggunakan paket `javax.sound.sampled` untuk mengalirkan audio dari perangkat input.

Kode

```
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.TargetDataLine;

public class Microphone {

    public static TargetDataLine get() throws Exception {
        AudioFormat format = new AudioFormat(16000, 16, 1, true, false);
        DataLine.Info datalineInfo = new DataLine.Info(TargetDataLine.class, format);

        TargetDataLine dataLine = (TargetDataLine) AudioSystem.getLine(datalineInfo);
        dataLine.open(format);

        return dataLine;
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Buat penerbit

Kode ini mengimplementasikan penerbit yang menerbitkan data audio dari aliran audio. Amazon Transcribe

### Kode

```
package com.amazonaws.transcribe;

import java.io.IOException;
import java.io.InputStream;
import java.io.UncheckedIOException;
import java.nio.ByteBuffer;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.atomic.AtomicLong;
import org.reactivestreams.Publisher;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.transcribestreaming.model.AudioEvent;
import software.amazon.awssdk.services.transcribestreaming.model.AudioStream;
```

```
import
software.amazon.awssdk.services.transcribestreaming.model.TranscribeStreamingException;

public class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;

    public AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {
        s.onSubscribe(new SubscriptionImpl(s, inputStream));
    }

    private class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;

        private SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
```

```
        AudioEvent audioEvent = audioEventFromBuffer(audioBuffer);
        subscriber.onNext(audioEvent);
    } else {
        subscriber.onComplete();
        break;
    }
    } while (demand.decrementAndGet() > 0);
} catch (TranscribeStreamingException e) {
    subscriber.onError(e);
}
});
}

@Override
public void cancel() {

}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

    int len = 0;
    try {
        len = inputStream.read(audioBytes);

        if (len <= 0) {
            audioBuffer = ByteBuffer.allocate(0);
        } else {
            audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
        }
    } catch (IOException e) {
        throw new UncheckedIOException(e);
    }

    return audioBuffer;
}

private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
```

```
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Buat klien dan mulai streaming

Dalam metode utama, buat objek permintaan, mulai aliran input audio dan buat instance penerbit dengan input audio.

Anda juga harus membuat [StartStreamTranscriptionResponseHandler](#) untuk menentukan cara menangani respons dari Amazon Transcribe.

Kemudian, gunakan `startStreamTranscription` metode ini untuk memulai streaming dua arah. `TranscribeStreamingAsyncClient`

### Impor

```
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.TargetDataLine;
import javax.sound.sampled.AudioInputStream;
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.transcribestreaming.TranscribeStreamingAsyncClient;
import
    software.amazon.awssdk.services.transcribestreaming.model.TranscribeStreamingException ;
import
    software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionRequest;
import software.amazon.awssdk.services.transcribestreaming.model.MediaEncoding;
import software.amazon.awssdk.services.transcribestreaming.model.LanguageCode;
import
    software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionResponseHandl
import software.amazon.awssdk.services.transcribestreaming.model.TranscriptEvent;
```

### Kode

```
public static void convertAudio(TranscribeStreamingAsyncClient client) throws
Exception {

    try {
```

```
        StartStreamTranscriptionRequest request =
StartStreamTranscriptionRequest.builder()
        .mediaEncoding(MediaEncoding.PCM)
        .languageCode(LanguageCode.EN_US)
        .mediaSampleRateHertz(16_000).build();

        TargetDataLine mic = Microphone.get();
        mic.start();

        AudioStreamPublisher publisher = new AudioStreamPublisher(new
AudioInputStream(mic));

        StartStreamTranscriptionResponseHandler response =
        StartStreamTranscriptionResponseHandler.builder().subscriber(e -> {
            TranscriptEvent event = (TranscriptEvent) e;
            event.transcript().results().forEach(r ->
r.alternatives().forEach(a -> System.out.println(a.transcript())));
        }).build();

        // Keeps Streaming until you end the Java program
        client.startStreamTranscription(request, publisher, response);

    } catch (TranscribeStreamingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Lihat [contoh lengkapnya](#) di GitHub.

## Informasi lain

- [Cara Kerjanya](#) di Panduan Amazon Transcribe Pengembang.
- [Memulai Dengan Streaming Audio](#) di Panduan Amazon Transcribe Pengembang.



## Contoh kode SDK for Java 2.x

Contoh kode dalam topik ini menunjukkan cara menggunakan AWS SDK for Java 2.x with AWS.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

Contoh lintas layanan adalah contoh aplikasi yang bekerja di beberapa Layanan AWS.

Contoh-contoh

- [Tindakan dan skenario menggunakan SDK for Java 2.x](#)
- [Contoh lintas layanan menggunakan SDK for Java 2.x](#)

## Tindakan dan skenario menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with Layanan AWS.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

Layanan

- [Contoh API Gateway menggunakan SDK for Java 2.x](#)
- [Contoh Application Auto Scaling menggunakan SDK for Java 2.x](#)
- [Contoh Application Recovery Controller menggunakan SDK for Java 2.x](#)
- [Contoh Aurora menggunakan SDK for Java 2.x](#)
- [Contoh Auto Scaling menggunakan SDK for Java 2.x](#)
- [Contoh Amazon Bedrock menggunakan SDK for Java 2.x](#)

- [Contoh Amazon Bedrock Runtime menggunakan SDK for Java 2.x](#)
- [CloudFront contoh menggunakan SDK for Java 2.x](#)
- [CloudWatch contoh menggunakan SDK for Java 2.x](#)
- [CloudWatch Contoh acara menggunakan SDK for Java 2.x](#)
- [CloudWatch Contoh log menggunakan SDK for Java 2.x](#)
- [Contoh Identitas Amazon Cognito menggunakan SDK for Java 2.x](#)
- [Contoh Penyedia Identitas Amazon Cognito menggunakan SDK for Java 2.x](#)
- [Amazon Comprehend contoh menggunakan SDK for Java 2.x](#)
- [Contoh DynamoDB menggunakan SDK for Java 2.x](#)
- [Contoh Amazon EC2 menggunakan SDK for Java 2.x](#)
- [Contoh Amazon ECS menggunakan SDK for Java 2.x](#)
- [Contoh Elastic Load Balancing menggunakan SDK for Java 2.x](#)
- [MediaStore contoh menggunakan SDK for Java 2.x](#)
- [OpenSearch Contoh layanan menggunakan SDK for Java 2.x](#)
- [EventBridge contoh menggunakan SDK for Java 2.x](#)
- [Contoh Forecast menggunakan SDK for Java 2.x](#)
- [AWS Glue contoh menggunakan SDK for Java 2.x](#)
- [HealthImaging contoh menggunakan SDK for Java 2.x](#)
- [Contoh IAM menggunakan SDK for Java 2.x](#)
- [AWS IoT contoh menggunakan SDK for Java 2.x](#)
- [AWS IoT data contoh menggunakan SDK for Java 2.x](#)
- [Contoh Amazon Keyspaces menggunakan SDK for Java 2.x](#)
- [Contoh Kinesis menggunakan SDK for Java 2.x](#)
- [AWS KMS contoh menggunakan SDK for Java 2.x](#)
- [Contoh Lambda menggunakan SDK for Java 2.x](#)
- [MediaConvert contoh menggunakan SDK for Java 2.x](#)
- [Contoh Migration Hub menggunakan SDK for Java 2.x](#)
- [Amazon Personalisasi contoh menggunakan SDK for Java 2.x](#)
- [Amazon Personalize Events contoh menggunakan SDK for Java 2.x](#)

- [Amazon Personalisasi contoh Runtime menggunakan SDK for Java 2.x](#)
- [Amazon Pinpoint contoh menggunakan SDK for Java 2.x](#)
- [Amazon Pinpoint SMS dan Voice API contoh menggunakan SDK for Java 2.x](#)
- [Contoh Amazon Polly menggunakan SDK for Java 2.x](#)
- [Contoh Amazon RDS menggunakan SDK for Java 2.x](#)
- [Contoh Amazon Redshift menggunakan SDK for Java 2.x](#)
- [Contoh Amazon Rekognition menggunakan SDK for Java 2.x](#)
- [Route 53 contoh pendaftaran domain menggunakan SDK for Java 2.x](#)
- [Contoh Amazon S3 menggunakan SDK for Java 2.x](#)
- [Contoh S3 Glacier menggunakan SDK for Java 2.x](#)
- [SageMaker contoh menggunakan SDK for Java 2.x](#)
- [Secrets Manager contoh menggunakan SDK for Java 2.x](#)
- [Amazon SES contoh menggunakan SDK for Java 2.x](#)
- [Amazon SES API v2 contoh menggunakan SDK for Java 2.x](#)
- [Contoh Amazon SNS menggunakan SDK for Java 2.x](#)
- [Contoh Amazon SQS menggunakan SDK for Java 2.x](#)
- [Contoh Step Functions menggunakan SDK for Java 2.x](#)
- [AWS STS contoh menggunakan SDK for Java 2.x](#)
- [AWS Support contoh menggunakan SDK for Java 2.x](#)
- [Contoh Systems Manager menggunakan SDK for Java 2.x](#)
- [Contoh Amazon Texttract menggunakan SDK for Java 2.x](#)
- [Contoh Amazon Transcribe menggunakan SDK for Java 2.x](#)

## Contoh API Gateway menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan API Gateway AWS SDK for Java 2.x with.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

## Tindakan

### Buat REST API

Contoh kode berikut menunjukkan cara membuat API Gateway REST API.

SDK for Java 2.x

#### Note

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

```
public static String createAPI(ApiGatewayClient apiGateway, String restApiId,
String restApiName) {

    try {
        CreateRestApiRequest request = CreateRestApiRequest.builder()
            .cloneFrom(restApiId)
            .description("Created using the Gateway Java API")
            .name(restApiName)
            .build();

        CreateRestApiResponse response = apiGateway.createRestApi(request);
        System.out.println("The id of the new api is " + response.id());
        return response.id();

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        return "";  
    }
```

- Untuk detail API, lihat [CreateRestApi](#) di Referensi AWS SDK for Java 2.x API.

## Hapus REST API

Contoh kode berikut menunjukkan cara menghapus API Gateway REST API.

### SDK for Java 2.x

#### Note

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

```
public static void deleteAPI(ApiGatewayClient apiGateway, String restApiId) {  
  
    try {  
        DeleteRestApiRequest request = DeleteRestApiRequest.builder()  
            .restApiId(restApiId)  
            .build();  
  
        apiGateway.deleteRestApi(request);  
        System.out.println("The API was successfully deleted");  
  
    } catch (ApiGatewayException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Untuk detail API, lihat [DeleteRestApi](#) di Referensi AWS SDK for Java 2.x API.

## Hapus penerapan

Contoh kode berikut menunjukkan cara menghapus penyebaran.

## SDK for Java 2.x

 Note

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

```
public static void deleteSpecificDeployment(ApiGatewayClient apiGateway, String
restApiId, String deploymentId) {

    try {
        DeleteDeploymentRequest request = DeleteDeploymentRequest.builder()
            .restApiId(restApiId)
            .deploymentId(deploymentId)
            .build();

        apiGateway.deleteDeployment(request);
        System.out.println("Deployment was deleted");

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DeleteDeployment](#) di Referensi AWS SDK for Java 2.x API.

## Menerapkan REST API

Contoh kode berikut menunjukkan cara menerapkan API Gateway REST API.

## SDK for Java 2.x

 Note

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

```
public static String createNewDeployment(ApiGatewayClient apiGateway, String
restApiId, String stageName) {

    try {
        CreateDeploymentRequest request = CreateDeploymentRequest.builder()
            .restApiId(restApiId)
            .description("Created using the AWS API Gateway Java API")
            .stageName(stageName)
            .build();

        CreateDeploymentResponse response =
apiGateway.createDeployment(request);
        System.out.println("The id of the deployment is " + response.id());
        return response.id();

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Untuk detail API, lihat [CreateDeployment](#) di Referensi AWS SDK for Java 2.x API.

## Contoh Application Auto Scaling menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan Application Auto Scaling AWS SDK for Java 2.x with.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

### Topik


- [Tindakan](#)

## Tindakan

Menonaktifkan sumber daya

Contoh kode berikut menunjukkan cara menonaktifkan sumber daya Application Auto Scaling.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingException;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DeregisterScalableTargetRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 */
```



```

* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class DisableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableId> <policyName>\s

            Where:
                tableId - The table Id value (for example, table/Music).\s
                policyName - The name of the policy (for example, $Music5-scaling-
policy).

            """;
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        ApplicationAutoScalingClient appAutoScalingClient =
ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        String tableId = args[0];
        String policyName = args[1];

        deletePolicy(appAutoScalingClient, policyName, tableWCUs, ns, tableId);
        verifyScalingPolicies(appAutoScalingClient, tableId, ns, tableWCUs);
        deregisterScalableTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
    }

    public static void deletePolicy(ApplicationAutoScalingClient
appAutoScalingClient, String policyName, ScalableDimension tableWCUs,
ServiceNamespace ns, String tableId) {
        try {

```

```
        DeleteScalingPolicyRequest delSPRequest =
DeleteScalingPolicyRequest.builder()
    .policyName(policyName)
    .scalableDimension(tableWCUs)
    .serviceNamespace(ns)
    .resourceId(tableId)
    .build();

        appAutoScalingClient.deleteScalingPolicy(delSPRequest);
        System.out.println(policyName + " was deleted successfully.");

    } catch (ApplicationAutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

// Verify that the scaling policy was deleted
public static void verifyScalingPolicies(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    DescribeScalingPoliciesRequest dscRequest =
DescribeScalingPoliciesRequest.builder()
    .scalableDimension(tableWCUs)
    .serviceNamespace(ns)
    .resourceId(tableId)
    .build();

    DescribeScalingPoliciesResponse response =
appAutoScalingClient.describeScalingPolicies(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(response);
}

public static void deregisterScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    try {
        DeregisterScalableTargetRequest targetRequest =
DeregisterScalableTargetRequest.builder()
    .scalableDimension(tableWCUs)
    .serviceNamespace(ns)
    .resourceId(tableId)
    .build();
```

```
        appAutoScalingClient.deregisterScalableTarget(targetRequest);
        System.out.println("The scalable target was deregistered.");

    } catch (ApplicationAutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
        .scalableDimension(tableWCUs)
        .serviceNamespace(ns)
        .resourceIds(tableId)
        .build();

    DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(response);
}
}
```

- Untuk detail API, lihat [DeleteScalingPolicy](#) di Referensi AWS SDK for Java 2.x API.

## Mendaftarkan sumber daya

Contoh kode berikut menunjukkan cara mendaftarkan sumber daya Application Auto Scaling.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
```

```
import
    software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingException;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResponse;
import software.amazon.awssdk.services.applicationautoscaling.model.PolicyType;
import
    software.amazon.awssdk.services.applicationautoscaling.model.PredefinedMetricSpecification;
import
    software.amazon.awssdk.services.applicationautoscaling.model.PutScalingPolicyRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.RegisterScalableTargetRequest;
import software.amazon.awssdk.services.applicationautoscaling.model.ScalingPolicy;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import software.amazon.awssdk.services.applicationautoscaling.model.MetricType;
import
    software.amazon.awssdk.services.applicationautoscaling.model.TargetTrackingScalingPolicyConfiguration;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
            <tableId> <roleARN> <policyName>\s
```

```
Where:
    tableId - The table Id value (for example, table/Music).
    roleARN - The ARN of the role that has ApplicationAutoScaling
permissions.
    policyName - The name of the policy to create.

""";

if (args.length != 3) {
    System.out.println(usage);
    System.exit(1);
}

System.out.println("This example registers an Amazon DynamoDB table, which
is the resource to scale.");
String tableId = args[0];
String roleARN = args[1];
String policyName = args[2];
ServiceNamespace ns = ServiceNamespace.DYNAMODB;
ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
ApplicationAutoScalingClient appAutoScalingClient =
ApplicationAutoScalingClient.builder()
    .region(Region.US_EAST_1)
    .build();

registerScalableTarget(appAutoScalingClient, tableId, roleARN, ns,
tableWCUs);
verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
configureScalingPolicy(appAutoScalingClient, tableId, ns, tableWCUs,
policyName);
}

public static void registerScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, String roleARN, ServiceNamespace ns,
ScalableDimension tableWCUs) {
    try {
        RegisterScalableTargetRequest targetRequest =
RegisterScalableTargetRequest.builder()
            .serviceNamespace(ns)
            .scalableDimension(tableWCUs)
            .resourceId(tableId)
            .roleARN(roleARN)
            .minCapacity(5)
```

```
        .maxCapacity(10)
        .build();

    appAutoScalingClient.registerScalableTarget(targetRequest);
    System.out.println("You have registered " + tableId);

} catch (ApplicationAutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}
}

// Verify that the target was created.
public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
        .scalableDimension(tableWCUs)
        .serviceNamespace(ns)
        .resourceIds(tableId)
        .build();

    DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(response);
}

// Configure a scaling policy.
public static void configureScalingPolicy(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs, String policyName) {
    // Check if the policy exists before creating a new one.
    DescribeScalingPoliciesResponse describeScalingPoliciesResponse =
appAutoScalingClient.describeScalingPolicies(DescribeScalingPoliciesRequest.builder()
        .serviceNamespace(ns)
        .resourceId(tableId)
        .scalableDimension(tableWCUs)
        .build());

    if (!describeScalingPoliciesResponse.scalingPolicies().isEmpty()) {
        // If policies exist, consider updating an existing policy instead of
        creating a new one.
    }
}
```

```
        System.out.println("Policy already exists. Consider updating it
instead.");
        List<ScalingPolicy> polList =
describeScalingPoliciesResponse.scalingPolicies();
        for (ScalingPolicy pol : polList) {
            System.out.println("Policy name:" +pol.policyName());
        }
    } else {
        // If no policies exist, proceed with creating a new policy.
        PredefinedMetricSpecification specification =
PredefinedMetricSpecification.builder()

.predefinedMetricType(MetricType.DYNAMO_DB_WRITE_CAPACITY_UTILIZATION)
        .build();

        TargetTrackingScalingPolicyConfiguration policyConfiguration =
TargetTrackingScalingPolicyConfiguration.builder()
            .predefinedMetricSpecification(specification)
            .targetValue(50.0)
            .scaleInCooldown(60)
            .scaleOutCooldown(60)
            .build();

        PutScalingPolicyRequest putScalingPolicyRequest =
PutScalingPolicyRequest.builder()
            .targetTrackingScalingPolicyConfiguration(policyConfiguration)
            .serviceNamespace(ns)
            .scalableDimension(tableWCUs)
            .resourceId(tableId)
            .policyName(policyName)
            .policyType(PolicyType.TARGET_TRACKING_SCALING)
            .build();

        try {
            appAutoScalingClient.putScalingPolicy(putScalingPolicyRequest);
            System.out.println("You have successfully created a scaling policy
for an Application Auto Scaling scalable target");
        } catch (ApplicationAutoScalingException e) {
            System.err.println("Error: " + e.awsErrorDetails().errorMessage());
        }
    }
}
}
```

- Untuk detail API, lihat [RegisterScalableTarget](#) di Referensi AWS SDK for Java 2.x API.

## Contoh Application Recovery Controller menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan kepada Anda bagaimana melakukan tindakan dan mengimplementasikan skenario umum AWS SDK for Java 2.x dengan menggunakan Application Recovery Controller.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

Tindakan

Dapatkan status kontrol perutean

Contoh kode berikut menunjukkan cara mendapatkan status kontrol routing Application Recovery Controller.

SDK for Java 2.x

### Note

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



```
public static GetRoutingControlStateResponse
getRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
    String routingControlArn) {
    // As a best practice, we recommend choosing a random cluster endpoint to
get or
    // set routing control states.
    // For more information, see
    // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
practices.html#route53-arc-best-practices.regional
    Collections.shuffle(clusterEndpoints);
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region())).build();
            return client.getRoutingControlState(
                GetRoutingControlStateRequest.builder()
                    .routingControlArn(routingControlArn).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
    return null;
}
```

- Untuk detail API, lihat [GetRoutingControlState](#) di Referensi AWS SDK for Java 2.x API.

## Perbarui status kontrol perutean

Contoh kode berikut menunjukkan cara memperbarui status kontrol routing Application Recovery Controller.

## SDK for Java 2.x

### Note

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

```

    public static UpdateRoutingControlStateResponse
updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
    String routingControlArn,
    String routingControlState) {
    // As a best practice, we recommend choosing a random cluster endpoint to
get or
    // set routing control states.
    // For more information, see
    // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
practices.html#route53-arc-best-practices.regional
    Collections.shuffle(clusterEndpoints);
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region()))
                .build();
            return client.updateRoutingControlState(
                UpdateRoutingControlStateRequest.builder()

.routingControlArn(routingControlArn).routingControlState(routingControlState).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
    return null;
}

```

- Untuk detail API, lihat [UpdateRoutingControlState](#) di Referensi AWS SDK for Java 2.x API.

## Contoh Aurora menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with Aurora.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.


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

Memulai

Halo Aurora

Contoh kode berikut ini menunjukkan cara mulai menggunakan Aurora.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        describeClusters(rdsClient);
        rdsClient.close();
    }

    public static void describeClusters(RdsClient rdsClient) {
        DescribeDBClustersIterable clustersIterable =
rdsClient.describeDBClustersPaginator();
        clustersIterable.stream()
            .flatMap(r -> r.dbClusters().stream())
            .forEach(cluster -> System.out
                .println("Database name: " + cluster.databaseName() + " Arn
= " + cluster.dbClusterArn()));
    }
}
```

```
}  
}
```

- Untuk detail API, lihat [DescribeDBClusters](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

### Membuat klaster DB

Contoh kode berikut menunjukkan cara membuat cluster Aurora DB.

### SDK for Java 2.x

#### Note

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

```
public static String createDBCluster(RdsClient rdsClient, String  
dbParameterGroupFamily, String dbName,  
    String dbClusterIdentifier, String userName, String password) {  
    try {  
        CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()  
            .databaseName(dbName)  
            .dbClusterIdentifier(dbClusterIdentifier)  
            .dbClusterParameterGroupName(dbParameterGroupFamily)  
            .engine("aurora-mysql")  
            .masterUsername(userName)  
            .masterUserPassword(password)  
            .build();  
  
        CreateDbClusterResponse response =  
            rdsClient.createDBCluster(clusterRequest);
```

```

        return response.dbCluster().dbClusterArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

```

- Untuk detail API, lihat [CreateDBCluster](#) di Referensi API AWS SDK for Java 2.x .

## Buat grup parameter klaster DB

Contoh kode berikut menunjukkan cara membuat kelompok parameter cluster Aurora DB.

SDK for Java 2.x

### Note

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

```

public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
        String dbParameterGroupFamily) {
    try {
        CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

    } catch (RdsException e) {

```

```
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [CreateDB ClusterParameterGroup](#) di AWS SDK for Java 2.x Referensi API.

## Membuat snapshot klaster DB

Contoh kode berikut menunjukkan cara membuat snapshot cluster Aurora DB.

SDK for Java 2.x

### Note

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

```
public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
    String dbSnapshotIdentifier) {
    try {
        CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
        System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [CreateDB ClusterSnapshot](#) di AWS SDK for Java 2.x Referensi API.

## Buat instans DB di klaster DB

Contoh kode berikut menunjukkan cara membuat instance DB di cluster Aurora DB.

### SDK for Java 2.x

#### Note

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

```
public static String createDBInstanceCluster(RdsClient rdsClient,
    String dbInstanceIdentifier,
    String dbInstanceClusterIdentifier,
    String instanceClass) {
    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .engine("aurora-mysql")
            .dbInstanceClass(instanceClass)
            .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Lihat detail API di [CreateDBInstance](#) dalam Referensi API AWS SDK for Java 2.x .

## Menghapus klaster DB

Contoh kode berikut menunjukkan cara menghapus cluster Aurora DB.

SDK for Java 2.x

### Note

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

```
public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
    try {
        DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .skipFinalSnapshot(true)
            .build();

        rdsClient.deleteDBCluster(deleteDbClusterRequest);
        System.out.println(dbInstanceClusterIdentifier + " was deleted!");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DeleteDBCluster](#) di Referensi API AWS SDK for Java 2.x .

## Menghapus grup parameter klaster DB

Contoh kode berikut menunjukkan cara menghapus kelompok parameter cluster Aurora DB.



## SDK for Java 2.x

 Note

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

```
public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
    throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(clusterDBARN) == 0) {
                    System.out.println(clusterDBARN + " still exists");
                    didFind = true;
                }
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.

                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }

        DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
```

```
        .builder()
        .dbClusterParameterGroupName(dbClusterGroupName)
        .build();

        rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
        System.out.println(dbClusterGroupName + " was deleted.");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DeleteDB ClusterParameterGroup](#) di Referensi AWS SDK for Java 2.x API.

## Hapus instans basis data

Contoh kode berikut menunjukkan cara menghapus instance Aurora DB.

### SDK for Java 2.x

#### Note

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

```
public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .deleteAutomatedBackups(true)
        .skipFinalSnapshot(true)
        .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
```

```
        System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Lihat detail API di [DeleteDBInstance](#) dalam Referensi API AWS SDK for Java 2.x .

## Menjelaskan grup parameter kluster DB

Contoh kode berikut menunjukkan bagaimana menggambarkan kelompok parameter cluster Aurora DB.

### SDK for Java 2.x

#### Note

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

```
public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
        DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .maxRecords(20)
            .build();

        List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
            .dbClusterParameterGroups();
        for (DBClusterParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbClusterParameterGroupName());
            System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
        }
    }
}
```

```

    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

```

- Untuk detail API, lihat [DescribeDB ClusterParameterGroups](#) di Referensi AWS SDK for Java 2.x API.

## Menjelaskan snapshot klaster DB

Contoh kode berikut menunjukkan bagaimana mendeskripsikan snapshot cluster Aurora DB.

SDK for Java 2.x

### Note

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

```

public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
    String dbInstanceClusterIdentifier) {
    try {
        boolean snapshotReady = false;
        String snapshotReadyStr;
        System.out.println("Waiting for the snapshot to become available.");

        DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .build();

        while (!snapshotReady) {
            DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);

```

```

        List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
        for (DBClusterSnapshot snapshot : snapshotList) {
            snapshotReadyStr = snapshot.status();
            if (snapshotReadyStr.contains("available")) {
                snapshotReady = true;
            } else {
                System.out.println(".");
                Thread.sleep(sleepTime * 5000);
            }
        }
    }

    System.out.println("The Snapshot is available!");

} catch (RdsException | InterruptedException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

```

- Untuk detail API, lihat [DescribeDB ClusterSnapshots](#) di Referensi AWS SDK for Java 2.x API.

## Menjelaskan klaster DB

Contoh kode berikut menunjukkan bagaimana menggambarkan cluster Aurora DB.

### SDK for Java 2.x

#### Note

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

```

public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLUSTERGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {

```

```

        dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
        .dbClusterParameterGroupName(dbCLusterGroupName)
        .build();
    } else {
        dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
        .dbClusterParameterGroupName(dbCLusterGroupName)
        .source("user")
        .build();
    }

DescribeDbClusterParametersResponse response = rdsClient
        .describeDBClusterParameters(dbParameterGroupsRequest);
List<Parameter> dbParameters = response.parameters();
String paraName;
for (Parameter para : dbParameters) {
    // Only print out information about either auto_increment_offset or
    // auto_increment_increment.
    paraName = para.parameterName();
    if ((paraName.compareTo("auto_increment_offset") == 0)
        || (paraName.compareTo("auto_increment_increment ") == 0)) {
        System.out.println("*** The parameter name is " + paraName);
        System.out.println("*** The parameter value is " +
para.parameterValue());
        System.out.println("*** The parameter data type is " +
para.dataType());
        System.out.println("*** The parameter description is " +
para.description());
        System.out.println("*** The parameter allowed values is " +
para.allowedValues());
    }
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

```

- Untuk detail API, lihat [DescribeDBClusters](#) di Referensi AWS SDK for Java 2.x API.

## Jelaskan instans basis data

Contoh kode berikut menunjukkan bagaimana menggambarkan instance Aurora DB.

### SDK for Java 2.x

#### Note

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

```
// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
            .dbClusterIdentifier(dbClusterIdentifier)
            .build();

        while (!instanceReady) {
            DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
            List<DBCluster> clusterList = response.dbClusters();
            for (DBCluster cluster : clusterList) {
                instanceReadyStr = cluster.status();
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
        System.out.println("Database cluster is available!");
    } catch (RdsException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}  
}
```

- Lihat detail API di [DescribeDBInstances](#) dalam Referensi API AWS SDK for Java 2.x .

Jelaskan versi mesin basis data

Contoh kode berikut menunjukkan bagaimana menggambarkan versi mesin database Aurora.

SDK for Java 2.x

**Note**

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

```
public static void describeDBEngines(RdsClient rdsClient) {  
    try {  
        DescribeDbEngineVersionsRequest engineVersionsRequest =  
DescribeDbEngineVersionsRequest.builder()  
            .engine("aurora-mysql")  
            .defaultOnly(true)  
            .maxRecords(20)  
            .build();  
  
        DescribeDbEngineVersionsResponse response =  
rdsClient.describeDBEngineVersions(engineVersionsRequest);  
        List<DBEngineVersion> engines = response.dbEngineVersions();  
  
        // Get all DBEngineVersion objects.  
        for (DBEngineVersion engineOb : engines) {  
            System.out.println("The name of the DB parameter group family for  
the database engine is "  
                + engineOb.dbParameterGroupFamily());  
            System.out.println("The name of the database engine " +  
engineOb.engine());  
            System.out.println("The version number of the database engine " +  
engineOb.engineVersion());  
        }  
    }  
}
```




```
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```

- Untuk detail API, lihat [DescribeDB EngineVersions](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan opsi untuk instans basis data

Contoh kode berikut menunjukkan bagaimana mendeskripsikan opsi untuk instance Aurora DB.

SDK for Java 2.x

 Note

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

```
public static void describeDBEngines(RdsClient rdsClient) {  
    try {  
        DescribeDbEngineVersionsRequest engineVersionsRequest =  
DescribeDbEngineVersionsRequest.builder()  
            .engine("aurora-mysql")  
            .defaultOnly(true)  
            .maxRecords(20)  
            .build();  
  
        DescribeDbEngineVersionsResponse response =  
rdsClient.describeDBEngineVersions(engineVersionsRequest);  
        List<DBEngineVersion> engines = response.dbEngineVersions();  
  
        // Get all DBEngineVersion objects.  
        for (DBEngineVersion engineOb : engines) {  
            System.out.println("The name of the DB parameter group family for  
the database engine is "  
                + engineOb.dbParameterGroupFamily());  
            System.out.println("The name of the database engine " +  
engineOb.engine());  
        }  
    }  
}
```

```

        System.out.println("The version number of the database engine " +
engine0b.engineVersion());
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

```

- Untuk detail API, lihat [DescribeOrderableDB InstanceOptions](#) di Referensi AWS SDK for Java 2.x API.

## Menjelaskan parameter dari grup parameter klaster DB

Contoh kode berikut menunjukkan bagaimana mendeskripsikan parameter dari kelompok parameter cluster Aurora DB.

### SDK for Java 2.x

#### Note

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

```

public static void describeDbClusterParameters(RdsClient rdsClient, String
dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .build();
        } else {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .source("user")

```

```

        .build();
    }

    DescribeDbClusterParametersResponse response = rdsClient
        .describeDBClusterParameters(dbParameterGroupsRequest);
    List<Parameter> dbParameters = response.parameters();
    String paraName;
    for (Parameter para : dbParameters) {
        // Only print out information about either auto_increment_offset or
        // auto_increment_increment.
        paraName = para.parameterName();
        if ((paraName.compareTo("auto_increment_offset") == 0)
            || (paraName.compareTo("auto_increment_increment ") == 0)) {
            System.out.println("*** The parameter name is " + paraName);
            System.out.println("*** The parameter value is " +
                para.parameterValue());
            System.out.println("*** The parameter data type is " +
                para.dataType());
            System.out.println("*** The parameter description is " +
                para.description());
            System.out.println("*** The parameter allowed values is " +
                para.allowedValues());
        }
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

```

- Untuk detail API, lihat [DescribeDB ClusterParameters](#) di Referensi AWS SDK for Java 2.x API.

## Memperbarui parameter dalam grup parameter klaster DB

Contoh kode berikut menunjukkan cara memperbarui parameter dalam kelompok parameter cluster Aurora DB.

## SDK for Java 2.x

 Note

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

```
public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
        DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .maxRecords(20)
            .build();

        List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
            .dbClusterParameterGroups();
        for (DBClusterParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbClusterParameterGroupName());
            System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ModifyDB ClusterParameterGroup](#) di AWS SDK for Java 2.x Referensi API.

## Skenario

### Memulai dengan klaster DB

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- 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 for Java 2.x

#### Note

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

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
 * services-use-secrets_RS.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition
 * by calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.
 * 2. Selects an engine family and creates a custom DB cluster parameter group
 * by invoking the describeDBClusterParameters method.
```

```

* 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups
* method.
* 4. Gets parameters in the group by invoking the describeDBClusterParameters
* method.
* 5. Modifies the auto_increment_offset parameter by invoking the
* modifyDbClusterParameterGroupRequest method.
* 6. Gets and displays the updated parameters.
* 7. Gets a list of allowed engine versions by invoking the
* describeDbEngineVersions method.
* 8. Creates an Aurora DB cluster database cluster that contains a MySQL
* database.
* 9. Waits for DB instance to be ready.
* 10. Gets a list of instance classes available for the selected engine.
* 11. Creates a database instance in the cluster.
* 12. Waits for DB instance to be ready.
* 13. Creates a snapshot.
* 14. Waits for DB snapshot to be ready.
* 15. Deletes the DB cluster.
* 16. Deletes the DB cluster group.
*/
public class AuroraScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
            "Usage:\n" +
            "    <dbClusterGroupName> <dbParameterGroupFamily>
<dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName>
<dbSnapshotIdentifier><secretName>"
            +
            "Where:\n" +
            "    dbClusterGroupName - The name of the DB cluster parameter
group. \n" +
            "    dbParameterGroupFamily - The DB cluster parameter group family
name (for example, aurora-mysql5.7). \n"
            +
            "    dbInstanceClusterIdentifier - The instance cluster identifier
value.\n" +
            "    dbInstanceIdentifier - The database instance identifier.\n" +
            "    dbName - The database name.\n" +
            "    dbSnapshotIdentifier - The snapshot identifier.\n" +
            "    secretName - The name of the AWS Secrets Manager secret that
contains the database credentials\n\n";

```

```
    ;

    if (args.length != 7) {
        System.out.println(usage);
        System.exit(1);
    }

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

    // Retrieve the database credentials using AWS Secrets Manager.
    Gson gson = new Gson();
    User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
    String username = user.getUsername();
    String userPassword = user.getPassword();

    Region region = Region.US_WEST_2;
    RdsClient rdsClient = RdsClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon Aurora example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Return a list of the available DB engines");
    describeDBEngines(rdsClient);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Create a custom parameter group");
    createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
dbParameterGroupFamily);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Get the parameter group");
```

```
describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get the parameters in the group");
describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Modify the auto_increment_offset parameter");
modifyDBClusterParas(rdsClient, dbClusterGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Display the updated parameter value");
describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of allowed engine versions");
getAllowedEngines(rdsClient, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Create an Aurora DB cluster database");
String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName,
dbName, dbInstanceClusterIdentifier,
    username, userPassword);
System.out.println("The ARN of the cluster is " + arnClusterVal);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Wait for DB instance to be ready");
waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get a list of instance classes available for the
selected engine");
String instanceClass = getListInstanceClasses(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Create a database instance in the cluster.");
```



```
String clusterDBARN = createDBInstanceCluster(rdsClient,
dbInstanceIdentifier, dbInstanceClusterIdentifier,
instanceClass);
System.out.println("The ARN of the database is " + clusterDBARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Wait for DB instance to be ready");
waitDBInstanceReady(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Create a snapshot");
createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Wait for DB snapshot to be ready");
waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Delete the DB instance");
deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Delete the DB cluster");
deleteCluster(rdsClient, dbInstanceClusterIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("16. Delete the DB cluster group");
deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The Scenario has successfully completed.");
System.out.println(DASHES);
rdsClient.close();
}
```

```
private static SecretsManagerClient getSecretClient() {
    Region region = Region.US_WEST_2;
    return SecretsManagerClient.builder()
        .region(region)

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

private static String getSecretValues(String secretName) {
    SecretsManagerClient secretClient = getSecretClient();
    GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
        .secretId(secretName)
        .build();

    GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
    return valueResponse.secretString();
}

public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
    throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(clusterDBARN) == 0) {
                    System.out.println(clusterDBARN + " still exists");
                    didFind = true;
                }
            }
            if ((index == listSize) && (!didFind)) {
```

```
        // Went through the entire list and did not find the
database ARN.
        isDataDel = true;
    }
    Thread.sleep(sleepTime * 1000);
    index++;
}
}

DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
    .builder()
    .dbClusterParameterGroupName(dbClusterGroupName)
    .build();

rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
System.out.println(dbClusterGroupName + " was deleted.");

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
    try {
        DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .skipFinalSnapshot(true)
            .build();

        rdsClient.deleteDBCluster(deleteDbClusterRequest);
        System.out.println(dbInstanceClusterIdentifier + " was deleted!");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
```

```
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .deleteAutomatedBackups(true)
                .skipFinalSnapshot(true)
                .build();

            DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
        String dbInstanceClusterIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                .dbClusterIdentifier(dbInstanceClusterIdentifier)
                .build();

            while (!snapshotReady) {
                DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
                List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                    }
                }
            }
        }
    }
}
```

```
        Thread.sleep(sleepTime * 5000);
    }
}

System.out.println("The Snapshot is available!");

} catch (RdsException | InterruptedException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}

}

public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
    String dbSnapshotIdentifier) {
    try {
        CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
        System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void waitDBInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();
```

```
String endpoint = "";
while (!instanceReady) {
    DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
    List<DBInstance> instanceList = response.dbInstances();
    for (DBInstance instance : instanceList) {
        instanceReadyStr = instance.dbInstanceStatus();
        if (instanceReadyStr.contains("available")) {
            endpoint = instance.endpoint().address();
            instanceReady = true;
        } else {
            System.out.print(".");
            Thread.sleep(sleepTime * 1000);
        }
    }
}
System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static String createDBInstanceCluster(RdsClient rdsClient,
String dbInstanceIdentifier,
String dbInstanceClusterIdentifier,
String instanceClass) {
    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .engine("aurora-mysql")
            .dbInstanceClass(instanceClass)
            .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();
    }
}
```

```
    } catch (RdsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static String getListInstanceClasses(RdsClient rdsClient) {
    try {
        DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest
            .builder()
            .engine("aurora-mysql")
            .maxRecords(20)
            .build();

        DescribeOrderableDbInstanceOptionsResponse response = rdsClient
            .describeOrderableDBInstanceOptions(optionsRequest);
        List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
        String instanceClass = "";
        for (OrderableDBInstanceOption instanceOption : instanceOptions) {
            instanceClass = instanceOption.dbInstanceClass();
            System.out.println("The instance class is " +
instanceOption.dbInstanceClass());
            System.out.println("The engine version is " +
instanceOption.engineVersion());
        }
        return instanceClass;
    } catch (RdsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
```

```
    try {
        DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
            .dbClusterIdentifier(dbClusterIdentifier)
            .build();

        while (!instanceReady) {
            DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
            List<DBCluster> clusterList = response.dbClusters();
            for (DBCluster cluster : clusterList) {
                instanceReadyStr = cluster.status();
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
        System.out.println("Database cluster is available!");
    } catch (RdsException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
    String dbClusterIdentifier, String userName, String password) {
    try {
        CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
            .databaseName(dbName)
            .dbClusterIdentifier(dbClusterIdentifier)
            .dbClusterParameterGroupName(dbParameterGroupFamily)
            .engine("aurora-mysql")
            .masterUsername(userName)
            .masterUserPassword(password)
            .build();

        CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
        return response.dbCluster().dbClusterArn();
    }
}
```



```
    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .engine("aurora-mysql")
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
        List<DBEngineVersion> dbEngines = response.dbEngineVersions();
        for (DBEngineVersion dbEngine : dbEngines) {
            System.out.println("The engine version is " +
dbEngine.engineVersion());
            System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Modify the auto_increment_offset parameter.
public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();
    }
}
```

```

        List<Parameter> paraList = new ArrayList<>();
        paraList.add(parameter1);
        ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
            .dbClusterParameterGroupName(dClusterGroupName)
            .parameters(paraList)
            .build();

        ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
        System.out.println(
            "The parameter group " + response.dbClusterParameterGroupName()
+ " was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDbClusterParameters(RdsClient rdsClient, String
dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .build();
        } else {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .source("user")
                .build();
        }

        DescribeDbClusterParametersResponse response = rdsClient
            .describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.

```

```
        paraName = para.parameterName();
        if ((paraName.compareTo("auto_increment_offset") == 0)
            || (paraName.compareTo("auto_increment_increment ") == 0)) {
            System.out.println("*** The parameter name is " + paraName);
            System.out.println("*** The parameter value is " +
para.parameterValue());
            System.out.println("*** The parameter data type is " +
para.dataType());
            System.out.println("*** The parameter description is " +
para.description());
            System.out.println("*** The parameter allowed values is " +
para.allowedValues());
        }
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
        DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .maxRecords(20)
            .build();

        List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
            .dbClusterParameterGroups();
        for (DBClusterParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbClusterParameterGroupName());
            System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
}

    public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
        String dbParameterGroupFamily) {
        try {
            CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .dbParameterGroupFamily(dbParameterGroupFamily)
                .description("Created by using the AWS SDK for Java")
                .build();

            CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
            System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                .engine("aurora-mysql")
                .defaultOnly(true)
                .maxRecords(20)
                .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
the database engine is "
                    + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
engineOb.engine());
            }
        }
    }
}
```

```
        System.out.println("The version number of the database engine " +
engineObj.engineVersion());
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [CreateDBCluster](#)
  - [dibuatB ClusterParameterGroup](#)
  - [dibuatB ClusterSnapshot](#)
  - [CreateDBInstance](#)
  - [DeleteDBCluster](#)
  - [DihapusB ClusterParameterGroup](#)
  - [DeleteDBInstance](#)
  - [DijelaskanB ClusterParameterGroups](#)
  - [DijelaskanB ClusterParameters](#)
  - [DijelaskanB ClusterSnapshots](#)
  - [DescribeDBClusters](#)
  - [DijelaskanB EngineVersions](#)
  - [DescribeDBInstances](#)
  - [DescribeOrderableDB InstanceOptions](#)
  - [ModifyDB ClusterParameterGroup](#)

## Contoh Auto Scaling menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan Auto Scaling AWS SDK for Java 2.x with.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

## Memulai

### Halo Auto Scaling

Contoh kode berikut menunjukkan cara memulai menggunakan Auto Scaling.

#### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    describeGroups(autoScalingClient);
}

public static void describeGroups(AutoScalingClient autoScalingClient) {
    DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups();
    List<AutoScalingGroup> groups = response.autoScalingGroups();
    groups.forEach(group -> {
        System.out.println("Group Name: " + group.autoScalingGroupName());
        System.out.println("Group ARN: " + group.autoScalingGroupARN());
    });
}
}
```

- Untuk detail API, lihat [DescribeAutoScalingGroups](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

### Membuat grup

Contoh kode berikut menunjukkan cara membuat grup Auto Scaling.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
    software.amazon.awssdk.services.autoscaling.model.CreateAutoScalingGroupRequest;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
    software.amazon.awssdk.services.autoscaling.model.LaunchTemplateSpecification;
import software.amazon.awssdk.services.autoscaling.waiters.AutoScalingWaiter;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <groupName> <launchTemplateName> <serviceLinkedRoleARN>
<vpcZoneId>

            Where:
                groupName - The name of the Auto Scaling group.
                launchTemplateName - The name of the launch template.\s
                vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
instances in the Auto Scaling group can be created.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
```



```
        .region(Region.US_EAST_1)
        .build();

        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
        autoScalingClient.close();
    }

    public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
        String groupName,
        String launchTemplateName,
        String vpcZoneId) {

        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(launchTemplateName)
                .build();

            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .availabilityZones("us-east-1a")
                .launchTemplate(templateSpecification)
                .maxSize(1)
                .minSize(1)
                .vpcZoneIdentifier(vpcZoneId)
                .build();

            autoScalingClient.createAutoScalingGroup(request);
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                .waitUntilGroupExists(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

```
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [CreateAutoScalingGroup](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus grup

Contoh kode berikut menunjukkan cara menghapus grup Auto Scaling.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
software.amazon.awssdk.services.autoscaling.model.DeleteAutoScalingGroupRequest;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <groupName>

                Where:
```

```

        groupName - The name of the Auto Scaling group.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String groupName = args[0];
    AutoScalingClient autoScalingClient = AutoScalingClient.builder()
        .region(Region.US_EAST_1)
        .build();

    deleteAutoScalingGroup(autoScalingClient, groupName);
    autoScalingClient.close();
}

public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build();

        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println("You successfully deleted " + groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [DeleteAutoScalingGroup](#) di Referensi AWS SDK for Java 2.x API.

## Menonaktifkan koleksi metrik untuk grup

Contoh kode berikut menunjukkan cara menonaktifkan koleksi CloudWatch metrik untuk grup Auto Scaling.

## SDK for Java 2.x

 Note

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

```
public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .build();

autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
        System.out.println("The disable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DisableMetricsCollection](#) di Referensi AWS SDK for Java 2.x API.

## Mengaktifkan pengumpulan metrik untuk grup

Contoh kode berikut menunjukkan cara mengaktifkan koleksi CloudWatch metrik untuk grup Auto Scaling.

## SDK for Java 2.x

 Note

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

```
public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [EnableMetricsCollection](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan informasi tentang grup

Contoh kode berikut menunjukkan cara mendapatkan informasi tentang grup Auto Scaling.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import software.amazon.awssdk.services.autoscaling.model.Instance;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAutoScalingInstances {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <groupName>

                Where:
                groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String groupName = args[0];
AutoScalingClient autoScalingClient = AutoScalingClient.builder()
    .region(Region.US_EAST_1)
    .build();

String instanceId = getAutoScaling(autoScalingClient, groupName);
System.out.println(instanceId);
autoScalingClient.close();
}

public static String getAutoScaling(AutoScalingClient autoScalingClient, String
groupName) {
    try {
        String instanceId = "";
        DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        DescribeAutoScalingGroupsResponse response = autoScalingClient
            .describeAutoScalingGroups(scalingGroupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group : groups) {
            System.out.println("The group name is " +
group.autoScalingGroupName());
            System.out.println("The group ARN is " +
group.autoScalingGroupARN());


            List<Instance> instances = group.instances();
            for (Instance instance : instances) {
                instanceId = instance.instanceId();
            }
        }
        return instanceId;
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Untuk detail API, lihat [DescribeAutoScalingGroups](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan informasi tentang instans

Contoh kode berikut menunjukkan cara mendapatkan informasi tentang instance Auto Scaling.

SDK for Java 2.x

 Note

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

```
public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
        .builder()
        .instanceIds(id)
        .build();

        DescribeAutoScalingInstancesResponse response = autoScalingClient
        .describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance : instances) {
            System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DescribeAutoScalingInstances](#) di Referensi AWS SDK for Java 2.x API.



## Dapatkan informasi tentang aktivitas penskalaan

Contoh kode berikut menunjukkan cara mendapatkan informasi tentang aktivitas Auto Scaling.

### SDK for Java 2.x

#### Note

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

```
public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
            .autoScalingGroupName(groupName)
            .maxRecords(10)
            .build();

        DescribeScalingActivitiesResponse response = autoScalingClient
            .describeScalingActivities(scalingActivitiesRequest);
        List<Activity> activities = response.activities();
        for (Activity activity : activities) {
            System.out.println("The activity Id is " + activity.activityId());
            System.out.println("The activity details are " +
activity.details());
        }


    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DescribeScalingActivities](#) di Referensi AWS SDK for Java 2.x API.

## Mengatur kapasitas yang diinginkan dari sebuah grup

Contoh kode berikut menunjukkan cara mengatur kapasitas yang diinginkan dari grup Auto Scaling.

## SDK for Java 2.x

 Note

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

```
public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
        .autoScalingGroupName(groupName)
        .desiredCapacity(2)
        .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");


    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [SetDesiredCapacity](#) di Referensi AWS SDK for Java 2.x API.

## Mengakhiri instance dalam grup

Contoh kode berikut menunjukkan cara menghentikan instance dalam grup Auto Scaling.

## SDK for Java 2.x

 Note

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

```
public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
        TerminateInstanceInAutoScalingGroupRequest.builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance " + instanceId);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [TerminateInstanceInAutoScalingGroup](#) di Referensi AWS SDK for Java 2.x API.

## Memperbarui grup

Contoh kode berikut menunjukkan cara memperbarui konfigurasi untuk grup Auto Scaling.

### SDK for Java 2.x

#### Note

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

```
public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
    String launchTemplateName) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
        LaunchTemplateSpecification.builder()
```

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

    UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
        .maxSize(3)
        .autoScalingGroupName(groupName)
        .launchTemplate(templateSpecification)
        .build();

    autoScalingClient.updateAutoScalingGroup(groupRequest);
    DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
        .waitUntilGroupInService(groupsRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("You successfully updated the auto scaling group " +
groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [UpdateAutoScalingGroup](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

### Membangun dan mengelola layanan yang tangguh

Contoh kode berikut menunjukkan cara membuat layanan web load-balanced yang mengembalikan rekomendasi buku, film, dan lagu. Contoh ini menunjukkan cara layanan tersebut merespons kegagalan, serta cara merestrukturisasi layanan agar lebih tangguh ketika terjadi kegagalan.

- Menggunakan grup Amazon EC2 Auto Scaling untuk membuat instans Amazon Elastic Compute Cloud (Amazon EC2) berdasarkan templat peluncuran dan menyimpan sejumlah instans dalam rentang yang ditentukan.
- Menangani dan mendistribusikan permintaan HTTP dengan Elastic Load Balancing.
- Memantau kondisi instans dalam grup Auto Scaling dan meneruskan permintaan hanya ke instans yang sehat.
- Menjalankan server web Python pada setiap instans EC2 untuk menangani permintaan HTTP. Server web merespons dengan memberikan rekomendasi dan melakukan pemeriksaan kondisi.
- Menyimulasikan layanan yang direkomendasikan dengan tabel Amazon DynamoDB.
- Kontrol respons server web terhadap permintaan dan pemeriksaan kesehatan dengan memperbarui AWS Systems Manager parameter.

## SDK for Java 2.x

### Note

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

Menjalankan skenario interaktif di prompt perintah.

```
public class Main {  
  
    public static final String fileName = "C:\\AWS\\resworkflow\\  
\\recommendations.json"; // Modify file location.  
    public static final String tableName = "doc-example-recommendation-service";  
    public static final String startScript = "C:\\AWS\\resworkflow\\  
\\server_startup_script.sh"; // Modify file location.  
    public static final String policyFile = "C:\\AWS\\resworkflow\\  
\\instance_policy.json"; // Modify file location.  
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\  
\\ssm_only_policy.json"; // Modify file location.  
    public static final String failureResponse = "doc-example-resilient-  
architecture-failure-response";  
    public static final String healthCheck = "doc-example-resilient-architecture-  
health-check";  
    public static final String templateName = "doc-example-resilience-template";  
    public static final String roleName = "doc-example-resilience-role";  
}
```

```
public static final String policyName = "doc-example-resilience-pol";
public static final String profileName = "doc-example-resilience-prof";

public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

public static final String targetGroupName = "doc-example-resilience-tg";
public static final String autoScalingGroupName = "doc-example-resilience-
group";
public static final String lbName = "doc-example-resilience-lb";
public static final String protocol = "HTTP";
public static final int port = 80;

public static final String DASHES = new String(new char[80]).replace("\0", "-");

public static void main(String[] args) throws IOException, InterruptedException
{
    Scanner in = new Scanner(System.in);
    Database database = new Database();
    AutoScaler autoScaler = new AutoScaler();
    LoadBalancer loadBalancer = new LoadBalancer();

    System.out.println(DASHES);
    System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("A - SETUP THE RESOURCES");
    System.out.println("Press Enter when you're ready to start deploying
resources.");
    in.nextLine();
    deploy(loadBalancer);
    System.out.println(DASHES);
    System.out.println(DASHES);
    System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    demo(loadBalancer);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("C - DELETE THE RESOURCES");
    System.out.println(""
```

This concludes the demo of how to build and manage a resilient service.

To keep things tidy and to avoid unwanted charges on your account, we can clean up all AWS resources

that were created for this demo.

```
""");
```

```
System.out.println("\n Do you want to delete the resources (y/n)? ");
```

```
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input
```

```
if (userInput.equals("y")) {
```

```
    // Delete resources here
```

```
    deleteResources(loadBalancer, autoScaler, database);
```

```
    System.out.println("Resources deleted.");
```

```
} else {
```

```
    System.out.println("""
```

```
        Okay, we'll leave the resources intact.
```

```
        Don't forget to delete them when you're done with them or you
```

might incur unexpected charges.

```
        """);
```

```
}
```

```
System.out.println(DASHES);
```

```
System.out.println(DASHES);
```

```
System.out.println("The example has completed. ");
```

```
System.out.println("\n Thanks for watching!");
```

```
System.out.println(DASHES);
```

```
}
```

```
// Deletes the AWS resources used in this example.
```

```
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler autoScaler, Database database)
```

```
    throws IOException, InterruptedException {
```

```
    loadBalancer.deleteLoadBalancer(lbName);
```

```
    System.out.println("*** Wait 30 secs for resource to be deleted");
```

```
    TimeUnit.SECONDS.sleep(30);
```

```
    loadBalancer.deleteTargetGroup(targetGroupName);
```

```
    autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
```

```
    autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
```

```
    autoScaler.deleteTemplate(templateName);
```

```
    database.deleteTable(tableName);
```

```
}
```

```

private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
    System.out.println(
        """
                For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
                to set up a load-balanced web service endpoint and explore
some ways to make it resilient
                against various kinds of failures.

                Some of the resources create by this demo are:
                \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
                \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
        """);

    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating and populating a DynamoDB table named " +
tableName);
    Database database = new Database();
    database.createTable(tableName, fileName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged credentials.
    """);

```



```
        The template also defines an IAM policy that each instance uses to
assume a role that grants
        permissions to access the DynamoDB recommendation table and Systems
Manager parameters
        that control the flow of the demo.
        """);
```

```
        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);
```

```
        System.out.println("""
            At this point, you have EC2 instances created. Once each instance
starts, it listens for
            HTTP requests. You can see these instances in the console or
continue with the demo.
            Press Enter when you're ready to continue.
            """);
```

```
        in.nextLine();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("""
            Creating an Elastic Load Balancing target group and load balancer.
The target group
```

```

        defines how the load balancer connects to instances. The load
balancer provides a
        single endpoint where clients connect and dispatches requests to
instances in the group.
        """);

    String vpcId = autoScaler.getDefaultVPC();
    List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
    System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
    String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
    String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
    autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
    System.out.println("Verifying access to the load balancer endpoint...");
    boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
    if (!wasSuccessful) {
        System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to "http://checkip.amazonaws.com"
        HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
        try {
            // Execute the request and get the response
            HttpResponse response = httpClient.execute(httpGet);

            // Read the response content.
            String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

            // Print the public IP address.
            System.out.println("Public IP Address: " + ipAddress);
            GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
            if (!groupInfo.isPortOpen()) {
                System.out.println("""
                    For this example to work, the default security group for
your default VPC must
                    allow access from this computer. You can either add it
automatically from this

```

```

        example or add it yourself using the AWS Management
Console.
        """);

        System.out.println(
            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
        System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
        String ans = in.nextLine();
        if ("y".equalsIgnoreCase(ans)) {
            autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
            System.out.println("Security group rule added.");
        } else {
            System.out.println("No security group rule added.");
        }
    }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
    } else if (wasSuccessful) {
        System.out.println("Your load balancer is ready. You can access it by
browsing to:");
        System.out.println("\t http://" + elbDnsName);
    } else {
        System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
        System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
        System.out.println("you can successfully make a GET request to the load
balancer.");
    }

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();

```

```
System.out.println("Read the ssm_only_policy.json file");
String ssmOnlyPolicy = readFileAsString(ssmJSON);

System.out.println("Resetting parameters to starting values for demo.");
paramHelper.reset();

System.out.println(
    """
        This part of the demonstration shows how to toggle
different parts of the system
        to create situations where the web service fails, and shows
how using a resilient
        architecture can keep the web service running in spite of
these failures.

        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
    """);
demoChoices(loadBalancer);

System.out.println(
    """
        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
    """);
paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

System.out.println(
    """
        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
    """);
demoChoices(loadBalancer);

System.out.println(
    """
        Instead of failing when the recommendation service fails,
the web service can return a static response.
    """);
```

```
        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
        """);
    paramHelper.put(paramHelper.failureResponse, "static");

    System.out.println("""
        Now, sending a GET request to the load balancer endpoint returns a
static response.
        The service still reports as healthy because health checks are still
shallow.
        """);
    demoChoices(loadBalancer);

    System.out.println("Let's reinstate the recommendation service.");
    paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

    System.out.println("""
        Let's also substitute bad credentials for one of the instances in
the target group so that it can't
        access the DynamoDB recommendation table. We will get an instance id
value.
        """);

    LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
    AutoScaler autoScaler = new AutoScaler();

    // Create a new instance profile based on badCredsProfileName.
    templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
    String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
    System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

    String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
    System.out.println("The association Id value is " + profileAssociationId);
    System.out.println("Replacing the profile for instance " + badInstanceId
        + " with a profile that contains bad credentials");
    autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

    System.out.println(
        """"
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
```

```
        depending on which instance is selected by the load
balancer.
        """);

        demoChoices(loadBalancer);

        System.out.println("""
            Let's implement a deep health check. For this demo, a deep health
check tests whether
            the web service can access the DynamoDB table that it depends on for
recommendations. Note that
            the deep health check is only for ELB routing and not for Auto
Scaling instance health.
            This kind of deep health check is not recommended for Auto Scaling
instance health, because it
            risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
        """);

        System.out.println("""
            By implementing deep health checks, the load balancer can detect
when one of the instances is failing
            and take that instance out of rotation.
        """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
            Now, checking target health indicates that the instance with bad
credentials
            is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
            instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
            the load balancer takes unhealthy instances out of its rotation.
        """);

        demoChoices(loadBalancer);

        System.out.println(
            ""
            Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
```

```

        instance is to terminate it and let the auto scaler start a
new instance to replace it.
        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
            Even while the instance is terminating and the new instance is
starting, sending a GET
            request to the web service continues to get a successful
recommendation response because
            the load balancer routes requests to the healthy instances. After
the replacement instance
            starts and reports as healthy, it is included in the load balancing
rotation.

            Note that terminating and replacing an instance typically takes
several minutes, during which time you
            can see the changing health check status until the new instance is
running and healthy.
        """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {

```

```
        System.out.println(i + ": " + actions[i]);
    }

    try {
        System.out.print("\nWhich action would you like to take? ");
        int choice = scanner.nextInt();
        System.out.println("-".repeat(88));

        switch (choice) {
            case 0 -> {
                System.out.println("Request:\n");
                System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                CloseableHttpClient httpClient =
HttpClientBuilder.createDefault();

                // Create an HTTP GET request to the ELB.
                HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);

                // Display the JSON response
                BufferedReader reader = new BufferedReader(
                    new
InputStreamReader(response.getEntity().getContent()));
                StringBuilder jsonResponse = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null) {
                    jsonResponse.append(line);
                }
                reader.close();

                // Print the formatted JSON response.
                System.out.println("Full Response:\n");
                System.out.println(jsonResponse.toString());

                // Close the HTTP client.
                httpClient.close();
            }
        }
    }
```



```

        case 1 -> {
            System.out.println("\nChecking the health of load balancer
targets:\n");
            List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
            for (TargetHealthDescription target : health) {
                System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
            }
            System.out.println("""
Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
        }
        case 2 -> {
            System.out.println("\nOkay, let's move on.");
            System.out.println("-".repeat(88));
            return; // Exit the method when choice is 2
        }
        default -> System.out.println("You must choose a value between
0-2. Please select again.");
    }

    } catch (java.util.InputMismatchException e) {
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}

```

Membuat kelas yang menggabungkan tindakan Penskalaan Otomatis dan Amazon EC2.

```
public class AutoScaler {
```

```
private static Ec2Client ec2Client;
private static AutoScalingClient autoScalingClient;
private static IamClient iamClient;

private static SsmClient ssmClient;

private IamClient getIAMClient() {
    if (iamClient == null) {
        iamClient = IamClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return iamClient;
}

private SsmClient getSSMClient() {
    if (ssmClient == null) {
        ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ssmClient;
}

private Ec2Client getEc2Client() {
    if (ec2Client == null) {
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
```

```

    * Terminates and instances in an EC2 Auto Scaling group. After an instance is
    * terminated, it can no longer be accessed.
    */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
        TerminateInstanceInAutoScalingGroupRequest
            .builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
    newInstanceProfileName, String profileAssociationId)
        throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
        iamInstanceProfile =
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
            .builder()
            .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
            // name.
            .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
        ReplaceIamInstanceProfileAssociationRequest
            .builder()
            .iamInstanceProfile(iamInstanceProfile)
            .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
            .build();
    }

```

```
try {
    getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
    // Handle the response as needed.
} catch (Ec2Exception e) {
    // Handle exceptions, log, or report the error.
    System.err.println("Error: " + e.getMessage());
}
System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
    newInstanceProfileName);
TimeUnit.SECONDS.sleep(15);
boolean instReady = false;
int tries = 0;

// Reboot after 60 seconds
while (!instReady) {
    if (tries % 6 == 0) {
        getEc2Client().rebootInstances(RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build());
        System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
    }
    tries++;
    try {
        TimeUnit.SECONDS.sleep(10);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
    List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
    for (InstanceInformation info : instanceInformationList) {
        if (info.instanceId().equals(instanceId)) {
            instReady = true;
            break;
        }
    }
}

SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
```

```

        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
            Collections.singletonList("cd / && sudo python3 server.py
80"))))
        .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(secGroupId)
            .cidrIp(ipAddress)
            .fromPort(Integer.parseInt(port))
            .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
            .builder()
            .instanceProfileName(profileName)
            .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.getInstanceProfile().getInstanceProfileName();
            System.out.println(name);
        }
    }

```

```
        RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .roleName(roleName)
        .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
        .roleName(roleName)
        .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
        .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(attachedPolicy.policyArn())
            .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");

    } catch (IamException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}

/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network, you
 * must instead specify a prefix list ID. You can also temporarily open the port
 * to
 * any IP address while running this example. If you do, be sure to remove
 * public
 * access when you're done.
 *
 */
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();
```

```

DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
    .filters(filter, filter1)
    .build();

DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
    .describeSecurityGroups(securityGroupsRequest);

String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
groupInfo.setGroupName(securityGroup);

for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
    System.out.println("Found security group: " + secGroup.groupId());

    for (IpPermission ipPermission : secGroup.ipPermissions()) {
        if (ipPermission.fromPort() == port) {
            System.out.println("Found inbound rule: " + ipPermission);
            for (IpRange ipRange : ipPermission.ipRanges()) {
                String cidrIp = ipRange.cidrIp();
                if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                    System.out.println(cidrIp + " is applicable");
                    portIsOpen = true;
                }
            }

            if (!ipPermission.prefixListIds().isEmpty()) {
                System.out.println("Prefix lList is applicable");
                portIsOpen = true;
            }

            if (!portIsOpen) {
                System.out
                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                    + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
            } else {
                break;
            }
        }
    }
}
}

```



```
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }

    groupInfo.setPortOpen(portIsOpen);
    return groupInfo;
}

/**
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

    // Get availability zones.
    software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
        .builder()
        .build();
```

```
        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
        .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();
```

```
        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
            .builder()
            .filters(defaultFilter)
            .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
            .name("vpc-id")
            .values(vpcId)
            .build();

        Filter azFilter = Filter.builder()
            .name("availability-zone")
            .values(availabilityZones)
            .build();

        Filter defaultForAZ = Filter.builder()
            .name("default-for-az")
            .values("true")
            .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
            .filters(vpcFilter, azFilter, defaultForAZ)
            .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();
```

```
DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
List<String> instanceIds = autoScalingGroup.instances().stream()
    .map(instance -> instance.instanceId())
    .collect(Collectors.toList());

String[] instanceIdArray = instanceIds.toArray(new String[0]);
for (String instanceId : instanceIdArray) {
    System.out.println("Instance ID: " + instanceId);
    return instanceId;
}
return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
    ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
    ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
    for (Policy policy : listPoliciesResponse.policies()) {
        if (policy.policyName().equals(policyName)) {
            // List the entities (users, groups, roles) that are attached to the
policy.

```

```

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
    .builder()
    .policyArn(policy.arn())
    .build();
ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
    .listEntitiesForPolicy(listEntitiesRequest);
if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
    || !listEntitiesResponse.policyRoles().isEmpty()) {
    // Detach the policy from any entities it is attached to.
    DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
        .policyArn(policy.arn())
        .roleName(roleName) // Specify the name of the IAM role
        .build();

    getIAMClient().detachRolePolicy(detachPolicyRequest);
    System.out.println("Policy detached from entities.");
}

// Now, you can delete the policy.
DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
    .policyArn(policy.arn())
    .build();

getIAMClient().deletePolicy(deletePolicyRequest);
System.out.println("Policy deleted successfully.");
break;
}
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);

```

```

        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .roleName(roleName) // Remove the extra dot here
                .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
            .instanceProfileName(InstanceProfile)
            .build();

        getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}

```

Membuat kelas yang menggabungkan tindakan Penyeimbangan Beban Elastis.

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.

```

```
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
        .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
```

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

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {
            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);
            if (statusCode == 200) {
                success = true;
            } else {
                retries--;
            }
        }
    }
}
```



```
        System.out.println("Got connection error from load balancer
endpoint, retrying...");
        TimeUnit.SECONDS.sleep(15);
    }
}

} catch (org.apache.http.conn.HttpHostConnectException e) {
    System.out.println(e.getMessage());
}

System.out.println("Status.." + success);
return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}

/*
```

```
* Creates an Elastic Load Balancing load balancer that uses the specified
* subnets
* and forwards requests to the specified target group.
*/
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
```

```

        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}

```

Membuat kelas yang menggunakan DynamoDB untuk menyimulasikan layanan yang direkomendasikan.

```

public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
    }
}

```

```

        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /**
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException {
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                .tableName(tableName)
                .attributeDefinitions(
                    AttributeDefinition.builder()
                        .attributeName("MediaType")
                        .attributeType(ScalarAttributeType.S)
                        .build(),

```

```

        AttributeDefinition.builder()
            .attributeName("ItemId")
            .attributeType(ScalarAttributeType.N)
            .build()
        .keySchema(
            KeySchemaElement.builder()
                .attributeName("MediaType")
                .keyType(KeyType.HASH)
                .build(),
            KeySchemaElement.builder()
                .attributeName("ItemId")
                .keyType(KeyType.RANGE)
                .build()
        ).provisionedThroughput(
            ProvisionedThroughput.builder()
                .readCapacityUnits(5L)
                .writeCapacityUnits(5L)
                .build()
        ).build();

    getDynamoDbClient().createTable(createTableRequest);
    System.out.println("Creating table " + tableName + "...");

    // Wait until the Amazon DynamoDB table is created.
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    WaiterResponse<DescribeTableResponse> waiterResponse =
    dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Table " + tableName + " created.");

    // Add records to the table.
    populateTable(fileName, tableName);
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB

```

```

// enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
    System.out.println("Added all records to the " + tableName);
}
}

```

Membuat kelas yang menggabungkan tindakan Systems Manager.

```

public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {

```

```
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
  - [AttachLoadBalancerTargetGroups](#)
  - [CreateAutoScalingGroup](#)
  - [CreateInstanceProfile](#)
  - [CreateLaunchTemplate](#)
  - [CreateListener](#)
  - [CreateLoadBalancer](#)
  - [CreateTargetGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DeleteInstanceProfile](#)
  - [DeleteLaunchTemplate](#)
  - [DeleteLoadBalancer](#)
  - [DeleteTargetGroup](#)
  - [DescribeAutoScalingGroups](#)

- [DescribeAvailabilityZones](#)
- [DescribeInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplaceInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

## Kelola grup dan instance

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat grup Auto Scaling Amazon EC2 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 for Java 2.x

### Note

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



```
/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a launch template. For more information, see the
 * following topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-
templates.html#create-launch-template
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
 * 13. Deletes the Auto Scaling group.
 */

public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <groupName> <launchTemplateName> <vpcZoneId>

            Where:
                groupName - The name of the Auto Scaling group.
                launchTemplateName - The name of the launch template.\s
    }
}
```

```
        vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
instances in the Auto Scaling group can be created.
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String groupName = args[0];
    String launchTemplateName = args[1];
    String vpcZoneId = args[2];
    AutoScalingClient autoScalingClient = AutoScalingClient.builder()
        .region(Region.US_EAST_1)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon EC2 Auto Scaling example
scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Create an Auto Scaling group named " + groupName);
    createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
    System.out.println(
        "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
    Thread.sleep(60000);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Get Auto Scale group Id value");
    String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupName);
    if (instanceId.compareTo("") == 0) {
        System.out.println("Error - no instance Id value");
        System.exit(1);
    } else {
        System.out.println("The instance Id value is " + instanceId);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
```

```
        System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
        describeAutoScalingInstance(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enable metrics collection " + instanceId);
        enableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update an Auto Scaling group to update max size to
3");
        updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Describe Auto Scaling groups");
        describeAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Describe account details");
        describeAccountLimits(autoScalingClient);
        System.out.println(
            "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Set desired capacity to 2");
        setDesiredCapacity(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get the two instance Id values and state");
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred for
the group");
        describeScalingActivities(autoScalingClient, groupName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group");
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                .autoScalingGroupName(groupName)
                .maxRecords(10)
                .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " + activity.activityId());
                System.out.println("The activity details are " +
activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

```
        System.exit(1);
    }
}

public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
            .autoScalingGroupName(groupName)
            .desiredCapacity(2)
            .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
String launchTemplateName,
String vpcZoneId) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .availabilityZones("us-east-1a")
            .launchTemplate(templateSpecification)
            .maxSize(1)
            .minSize(1)
            .vpcZoneIdentifier(vpcZoneId)
            .build();

        autoScalingClient.createAutoScalingGroup(request);
    }
}
```

```

        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
        .waitUntilGroupExists(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Auto Scaling Group created");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
        .builder()
        .instanceIds(id)
        .build();

        DescribeAutoScalingInstancesResponse response = autoScalingClient
        .describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance : instances) {
            System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {

```

```
    try {
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .maxRecords(10)
            .build();

        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group : groups) {
            System.out.println("*** The service to use for the health checks: "
+ group.healthCheckType());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        String instanceId = "";
        DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        DescribeAutoScalingGroupsResponse response = autoScalingClient
            .describeAutoScalingGroups(scalingGroupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group : groups) {
            System.out.println("The group name is " +
group.autoScalingGroupName());
            System.out.println("The group ARN is " +
group.autoScalingGroupARN());
            List<Instance> instances = group.instances();

            for (Instance instance : instances) {
                instanceId = instance.instanceId();
                System.out.println("The instance id is " + instanceId);
            }
        }
    }
}
```

```
        System.out.println("The lifecycle state is " +
instance.lifecycleState());
    }
}

    return instanceId;
} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .build();

        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
```



```
        System.out.println("The disable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAccountLimits(AutoScalingClient autoScalingClient) {
    try {
        DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
        System.out.println("The max number of auto scaling groups is " +
response.maxNumberOfAutoScalingGroups());
        System.out.println("The current number of auto scaling groups is " +
response.numberofAutoScalingGroups());

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
    String launchTemplateName) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
            .maxSize(3)
            .autoScalingGroupName(groupName)
            .launchTemplate(templateSpecification)
            .build();

        autoScalingClient.updateAutoScalingGroup(groupRequest);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
```

```
        .autoScalingGroupNames(groupName)
        .build();

    WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
        .waitUntilGroupInService(groupsRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("You successfully updated the auto scaling group " +
groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance " + instanceId);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build();

        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
```

```
        System.out.println("You successfully deleted " + groupName);
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [CreateAutoScalingGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DescribeAutoScalingGroups](#)
  - [DescribeAutoScalingInstances](#)
  - [DescribeScalingActivities](#)
  - [DisableMetricsCollection](#)
  - [EnableMetricsCollection](#)
  - [SetDesiredCapacity](#)
  - [TerminateInstanceInAutoScalingGroup](#)
  - [UpdateAutoScalingGroup](#)

## Contoh Amazon Bedrock menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x With Amazon Bedrock.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

### Topik

- [Tindakan](#)

## Tindakan

Dapatkan detail tentang model fondasi Amazon Bedrock

Contoh kode berikut menunjukkan cara mendapatkan detail tentang model pondasi Amazon Bedrock.

SDK for Java 2.x

### Note

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

Dapatkan detail tentang model foundation menggunakan klien Amazon Bedrock sinkron.

```
/**
 * Get details about an Amazon Bedrock foundation model.
 *
 * @param bedrockClient The service client for accessing Amazon Bedrock.
 * @param modelIdentifier The model identifier.
 * @return An object containing the foundation model's details.
 */
public static FoundationModelDetails getFoundationModel(BedrockClient
bedrockClient, String modelIdentifier) {
    try {
        GetFoundationModelResponse response = bedrockClient.getFoundationModel(
            r -> r.modelIdentifier(modelIdentifier)
        );

        FoundationModelDetails model = response.modelDetails();

        System.out.println(" Model ID:                " + model.modelId());
        System.out.println(" Model ARN:                " +
model.modelArn());
        System.out.println(" Model Name:                " +
model.modelName());
        System.out.println(" Provider Name:            " +
model.providerName());
    }
}
```

```

        System.out.println(" Lifecycle status: " +
model.modelLifecycle().statusAsString());
        System.out.println(" Input modalities: " +
model.inputModalities());
        System.out.println(" Output modalities: " +
model.outputModalities());
        System.out.println(" Supported customizations: " +
model.customizationsSupported());
        System.out.println(" Supported inference types: " +
model.inferenceTypesSupported());
        System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

        return model;

    } catch (ValidationException e) {
        throw new IllegalArgumentException(e.getMessage());
    } catch (SdkException e) {
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}

```

Dapatkan detail tentang model foundation menggunakan klien Amazon Bedrock asinkron.

```

/**
 * Get details about an Amazon Bedrock foundation model.
 *
 * @param bedrockClient The async service client for accessing Amazon Bedrock.
 * @param modelIdentifier The model identifier.
 * @return An object containing the foundation model's details.
 */
public static FoundationModelDetails getFoundationModel(BedrockAsyncClient
bedrockClient, String modelIdentifier) {
    try {
        CompletableFuture<GetFoundationModelResponse> future =
bedrockClient.getFoundationModel(
            r -> r.modelIdentifier(modelIdentifier)
        );

        FoundationModelDetails model = future.get().modelDetails();
    }
}

```

```

        System.out.println(" Model ID: " + model.modelId());
        System.out.println(" Model ARN: " +
model.modelArn());
        System.out.println(" Model Name: " +
model.modelName());
        System.out.println(" Provider Name: " +
model.providerName());
        System.out.println(" Lifecycle status: " +
model.modelLifecycle().statusAsString());
        System.out.println(" Input modalities: " +
model.inputModalities());
        System.out.println(" Output modalities: " +
model.outputModalities());
        System.out.println(" Supported customizations: " +
model.customizationsSupported());
        System.out.println(" Supported inference types: " +
model.inferenceTypesSupported());
        System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

        return model;

    } catch (ExecutionException e) {
        if (e.getMessage().contains("ValidationException")) {
            throw new IllegalArgumentException(e.getMessage());
        } else {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}

```

- Untuk detail API, lihat [GetFoundationModel](#) di Referensi AWS SDK for Java 2.x API.

Daftar model yayasan Amazon Bedrock yang tersedia

Contoh kode berikut menunjukkan cara membuat daftar model dasar Amazon Bedrock yang tersedia.

## SDK for Java 2.x

 Note

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

Buat daftar model foundation Amazon Bedrock yang tersedia menggunakan klien Amazon Bedrock sinkron.

```
/**
 * Lists Amazon Bedrock foundation models that you can use.
 * You can filter the results with the request parameters.
 *
 * @param bedrockClient The service client for accessing Amazon Bedrock.
 * @return A list of objects containing the foundation models' details
 */
public static List<FoundationModelSummary> listFoundationModels(BedrockClient
bedrockClient) {

    try {
        ListFoundationModelsResponse response =
bedrockClient.listFoundationModels(r -> {});

        List<FoundationModelSummary> models = response.modelSummaries();

        if (models.isEmpty()) {
            System.out.println("No available foundation models in " +
region.toString());
        } else {
            for (FoundationModelSummary model : models) {
                System.out.println("Model ID: " + model.modelId());
                System.out.println("Provider: " + model.providerName());
                System.out.println("Name:      " + model.modelName());
                System.out.println();
            }
        }

        return models;

    } catch (SdkClientException e) {
```

```

        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}

```

Buat daftar model foundation Amazon Bedrock yang tersedia menggunakan klien Amazon Bedrock asinkron.

```

/**
 * Lists Amazon Bedrock foundation models that you can use.
 * You can filter the results with the request parameters.
 *
 * @param bedrockClient The async service client for accessing Amazon Bedrock.
 * @return A list of objects containing the foundation models' details
 */
public static List<FoundationModelSummary>
listFoundationModels(BedrockAsyncClient bedrockClient) {
    try {
        CompletableFuture<ListFoundationModelsResponse> future =
bedrockClient.listFoundationModels(r -> {});

        List<FoundationModelSummary> models = future.get().modelSummaries();

        if (models.isEmpty()) {
            System.out.println("No available foundation models in " +
region.toString());
        } else {
            for (FoundationModelSummary model : models) {
                System.out.println("Model ID: " + model.modelId());
                System.out.println("Provider: " + model.providerName());
                System.out.println("Name:      " + model.modelName());
                System.out.println();
            }
        }

        return models;
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    } catch (ExecutionException e) {

```



```
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}
```

- Untuk detail API, lihat [ListFoundationModels](#) di Referensi AWS SDK for Java 2.x API.

## Contoh Amazon Bedrock Runtime menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan Runtime AWS SDK for Java 2.x with Amazon Bedrock.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

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

### Tindakan

#### Pembuatan gambar dengan Amazon Titan Image Generator G1

Contoh kode berikut menunjukkan cara memanggil model Amazon Titan Image Generator G1 di Amazon Bedrock untuk pembuatan gambar.

## SDK for Java 2.x

 Note

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

Secara asinkron memanggil model Amazon Titan Image Generator G1 untuk menghasilkan gambar.

```
/**
 * Invokes the Amazon Titan image generation model to create an image using the
 * input
 * provided in the request body.
 *
 * @param prompt The prompt that you want Amazon Titan to use for image
 * generation.
 * @param seed The random noise seed for image generation (Range: 0 to
 * 2147483647).
 * @return A Base64-encoded string representing the generated image.
 */
public static String invokeTitanImage(String prompt, long seed) {
    /**
     * The different model providers have individual request and response
     formats.
     * For the format, ranges, and default values for Titan Image models refer
     to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
     titan-
     * image.html
     */
    String titanImageModelId = "amazon.titan-image-generator-v1";

    BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
        .region(Region.US_EAST_1)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    var textToImageParams = new JSONObject().put("text", prompt);

    var imageGenerationConfig = new JSONObject()
```

```
        .put("numberOfImages", 1)
        .put("quality", "standard")
        .put("cfgScale", 8.0)
        .put("height", 512)
        .put("width", 512)
        .put("seed", seed);

JSONObject payload = new JSONObject()
    .put("taskType", "TEXT_IMAGE")
    .put("textToImageParams", textToImageParams)
    .put("imageGenerationConfig", imageGenerationConfig);

InvokeModelRequest request = InvokeModelRequest.builder()
    .body(SdkBytes.fromUtf8String(payload.toString()))
    .modelId(titanImageModelId)
    .contentType("application/json")
    .accept("application/json")
    .build();

CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request)
    .whenComplete((response, exception) -> {
        if (exception != null) {
            System.out.println("Model invocation failed: " + exception);
        }
    });

String base64ImageData = "";
try {
    InvokeModelResponse response = completableFuture.get();
    JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
    base64ImageData = responseBody
        .getJSONArray("images")
        .getString(0);

} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
    System.err.println(e.getMessage());
} catch (ExecutionException e) {
    System.err.println(e.getMessage());
}

return base64ImageData;
```

```
}

```

Minta model Amazon Titan Image Generator G1 untuk menghasilkan gambar.

```

/**
 * Invokes the Amazon Titan image generation model to create an image using
the
 * input
 * provided in the request body.
 *
 * @param prompt The prompt that you want Amazon Titan to use for image
 *               generation.
 * @param seed   The random noise seed for image generation (Range: 0 to
 *               2147483647).
 * @return A Base64-encoded string representing the generated image.
 */
public static String invokeTitanImage(String prompt, long seed) {
    /**
     * The different model providers have individual request and
response formats.
     * For the format, ranges, and default values for Titan Image models
refer to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-titan-
image.html
     */
    String titanImageModelId = "amazon.titan-image-generator-v1";

    BedrockRuntimeClient client = BedrockRuntimeClient.builder()
        .region(Region.US_EAST_1)

        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    var textToImageParams = new JSONObject().put("text", prompt);

    var imageGenerationConfig = new JSONObject()
        .put("numberOfImages", 1)
        .put("quality", "standard")
        .put("cfgScale", 8.0)
        .put("height", 512)
        .put("width", 512)

```

```
        .put("seed", seed);

    JSONObject payload = new JSONObject()
        .put("taskType", "TEXT_IMAGE")
        .put("textToImageParams", textToImageParams)
        .put("imageGenerationConfig",
imageGenerationConfig);

    InvokeModelRequest request = InvokeModelRequest.builder()
        .body(SdkBytes.fromUtf8String(payload.toString()))
        .modelId(titanImageModelId)
        .contentType("application/json")
        .accept("application/json")
        .build();

    InvokeModelResponse response = client.invokeModel(request);

    JSONObject responseBody = new
JSONObject(response.body().asUtf8String());

    String base64ImageData = responseBody
        .getJSONArray("images")
        .getString(0);

    return base64ImageData;
}
```

- Untuk detail API, lihat [InvokeModel](#) di Referensi AWS SDK for Java 2.x API.

## Pembuatan gambar dengan Stability.ai Stable Diffusion XL

Contoh kode berikut menunjukkan cara menjalankan model Stability.ai Stable Diffusion XL di Amazon Bedrock untuk pembuatan gambar.

### SDK for Java 2.x

#### Note

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

Secara asinkron memanggil model dasar Stability.ai Stable Diffusion XL untuk menghasilkan gambar.

```
/**
 * Asynchronously invokes the Stability.ai Stable Diffusion XL model to create
 * an image based on the provided input.
 *
 * @param prompt      The prompt that guides the Stable Diffusion model.
 * @param seed        The random noise seed for image generation (use 0 or omit
 *                    for a random seed).
 * @param stylePreset The style preset to guide the image model towards a
 *                    specific style.
 * @return A Base64-encoded string representing the generated image.
 */
public static String invokeStableDiffusion(String prompt, long seed, String
stylePreset) {
    /**
     * The different model providers have individual request and response
     formats.
     * For the format, ranges, and available style_presets of Stable Diffusion
     * models refer to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
     stability-diffusion.html
     */

    String stableDiffusionModelId = "stability.stable-diffusion-xl";

    BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
        .region(Region.US_EAST_1)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    JSONArray wrappedPrompt = new JSONArray().put(new JSONObject().put("text",
prompt));
    JSONObject payload = new JSONObject()
        .put("text_prompts", wrappedPrompt)
        .put("seed", seed);

    if (stylePreset != null && !stylePreset.isEmpty()) {
        payload.put("style_preset", stylePreset);
    }

    InvokeModelRequest request = InvokeModelRequest.builder()
```

```

        .body(SdkBytes.fromUtf8String(payload.toString()))
        .modelId(stableDiffusionModelId)
        .contentType("application/json")
        .accept("application/json")
        .build();

    CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                System.out.println("Model invocation failed: " + exception);
            }
        });

    String base64ImageData = "";
    try {
        InvokeModelResponse response = completableFuture.get();
        JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
        base64ImageData = responseBody
            .getJSONArray("artifacts")
            .getJSONObject(0)
            .getString("base64");

    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
    } catch (ExecutionException e) {
        System.err.println(e.getMessage());
    }

    return base64ImageData;
}

```

Memanggil model dasar Stability.ai Stable Diffusion XL untuk menghasilkan gambar.

```

/**
 * Invokes the Stability.ai Stable Diffusion XL model to create an image
based
 * on the provided input.
 *
 * @param prompt The prompt that guides the Stable Diffusion model.

```

```

    * @param seed          The random noise seed for image generation (use 0 or
omit
    *                      for a random seed).
    * @param stylePreset The style preset to guide the image model towards a
    *                      specific style.
    * @return A Base64-encoded string representing the generated image.
    */
    public static String invokeStableDiffusion(String prompt, long seed, String
stylePreset) {
        /*
        * The different model providers have individual request and
response formats.
        * For the format, ranges, and available style_presets of Stable
Diffusion
        * models refer to:
        * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-stability-diffusion.html
        */

        String stableDiffusionModelId = "stability.stable-diffusion-xl";

        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
            .region(Region.US_EAST_1)

            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        JSONArray wrappedPrompt = new JSONArray().put(new
JSONObject().put("text", prompt));

        JSONObject payload = new JSONObject()
            .put("text_prompts", wrappedPrompt)
            .put("seed", seed);

        if (!(stylePreset == null || stylePreset.isEmpty())) {
            payload.put("style_preset", stylePreset);
        }

        InvokeModelRequest request = InvokeModelRequest.builder()
            .body(SdkBytes.fromUtf8String(payload.toString()))
            .modelId(stableDiffusionModelId)
            .contentType("application/json")
            .accept("application/json")
            .build();

```



```
        InvokeModelResponse response = client.invokeModel(request);

        JSONObject responseBody = new
        JSONObject(response.body().asUtf8String());

        String base64ImageData = responseBody
            .getJSONArray("artifacts")
            .getJSONObject(0)
            .getString("base64");

        return base64ImageData;
    }
}
```

- Untuk detail API, lihat [InvokeModel](#) di Referensi AWS SDK for Java 2.x API.

## Pembuatan teks dengan AI21 Labs Jurassic-2

Contoh kode berikut menunjukkan cara menjalankan model AI21 Labs Jurassic-2 di Amazon Bedrock untuk pembuatan teks.

### SDK for Java 2.x

#### Note

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

Secara asinkron memanggil model dasar AI21 Labs Jurassic-2 untuk menghasilkan teks.

```
/**
 * Asynchronously invokes the AI21 Labs Jurassic-2 model to run an inference
 * based on the provided input.
 *
 * @param prompt The prompt that you want Jurassic to complete.
 * @return The inference response generated by the model.
 */
public static String invokeJurassic2(String prompt) {
    /**
```

```
    * The different model providers have individual request and response
formats.
    * For the format, ranges, and default values for Anthropic Claude, refer
to:
    * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
claude.html
    */

String jurassic2ModelId = "ai21.j2-mid-v1";

BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

String payload = new JSONObject()
    .put("prompt", prompt)
    .put("temperature", 0.5)
    .put("maxTokens", 200)
    .toString();

InvokeModelRequest request = InvokeModelRequest.builder()
    .body(SdkBytes.fromUtf8String(payload))
    .modelId(jurassic2ModelId)
    .contentType("application/json")
    .accept("application/json")
    .build();

CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request)
    .whenComplete((response, exception) -> {
        if (exception != null) {
            System.out.println("Model invocation failed: " + exception);
        }
    });

String generatedText = "";
try {
    InvokeModelResponse response = completableFuture.get();
    JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
    generatedText = responseBody
        .getJSONArray("completions")
        .getJSONObject(0)
```

```

        .getJSONObject("data")
        .getString("text");

    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
    } catch (ExecutionException e) {
        System.err.println(e.getMessage());
    }

    return generatedText;
}

```

Gunakan model dasar AI21 Labs Jurassic-2 untuk menghasilkan teks.

```

/**
 * Invokes the AI21 Labs Jurassic-2 model to run an inference based on the
 * provided input.
 *
 * @param prompt The prompt for Jurassic to complete.
 * @return The generated response.
 */
public static String invokeJurassic2(String prompt) {
    /**
     * The different model providers have individual request and
response formats.
     * For the format, ranges, and default values for AI21 Labs
Jurassic-2, refer
     * to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-jurassic2.html
     */

    String jurassic2ModelId = "ai21.j2-mid-v1";

    BedrockRuntimeClient client = BedrockRuntimeClient.builder()
        .region(Region.US_EAST_1)

        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    String payload = new JSONObject()

```

```
        .put("prompt", prompt)
        .put("temperature", 0.5)
        .put("maxTokens", 200)
        .toString();

    InvokeModelRequest request = InvokeModelRequest.builder()
        .body(SdkBytes.fromUtf8String(payload))
        .modelId(jurassic2ModelId)
        .contentType("application/json")
        .accept("application/json")
        .build();

    InvokeModelResponse response = client.invokeModel(request);

    JSONObject responseBody = new
    JSONObject(response.body().asUtf8String());

    String generatedText = responseBody
        .getJSONArray("completions")
        .getJSONObject(0)
        .getJSONObject("data")
        .getString("text");

    return generatedText;
}
```

- Untuk detail API, lihat [InvokeModel](#) di Referensi AWS SDK for Java 2.x API.

## Pembuatan teks dengan Anthropic Claude 2

Contoh kode berikut menunjukkan cara memanggil model Anthropic Claude 2 di Amazon Bedrock untuk pembuatan teks.

### SDK for Java 2.x

#### Note

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

## Secara asinkron memanggil model dasar Anthropic Claude 2 untuk menghasilkan teks.

```
/**
 * Asynchronously invokes the Anthropic Claude 2 model to run an inference based
 * on the provided input.
 *
 * @param prompt The prompt that you want Claude to complete.
 * @return The inference response from the model.
 */
public static String invokeClaude(String prompt) {
    /**
     * The different model providers have individual request and response
     formats.
     * For the format, ranges, and default values for Anthropic Claude, refer
     to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
     claude.html
     */

    String claudeModelId = "anthropic.claude-v2";

    // Claude requires you to enclose the prompt as follows:
    String enclosedPrompt = "Human: " + prompt + "\n\nAssistant:";

    BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
        .region(Region.US_EAST_1)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    String payload = new JSONObject()
        .put("prompt", enclosedPrompt)
        .put("max_tokens_to_sample", 200)
        .put("temperature", 0.5)
        .put("stop_sequences", List.of("\n\nHuman:"))
        .toString();

    InvokeModelRequest request = InvokeModelRequest.builder()
        .body(SdkBytes.fromUtf8String(payload))
        .modelId(claudeModelId)
        .contentType("application/json")
        .accept("application/json")
        .build();
}
```

```

    CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request)
    .whenComplete((response, exception) -> {
        if (exception != null) {
            System.out.println("Model invocation failed: " + exception);
        }
    });

String generatedText = "";
try {
    InvokeModelResponse response = completableFuture.get();
    JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
    generatedText = responseBody.getString("completion");
} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
    System.err.println(e.getMessage());
} catch (ExecutionException e) {
    System.err.println(e.getMessage());
}

return generatedText;
}

```

Gunakan model dasar Anthropic Claude 2 untuk menghasilkan teks.

```

/**
 * Invokes the Anthropic Claude 2 model to run an inference based on the
 * provided input.
 *
 * @param prompt The prompt for Claude to complete.
 * @return The generated response.
 */
public static String invokeClaude(String prompt) {
    /**
     * The different model providers have individual request and
response formats.
     * For the format, ranges, and default values for Anthropic Claude,
refer to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-claude.html
     */
}

```

```
String claudeModelId = "anthropic.claude-v2";

// Claude requires you to enclose the prompt as follows:
String enclosedPrompt = "Human: " + prompt + "\n\nAssistant:";

BedrockRuntimeClient client = BedrockRuntimeClient.builder()
    .region(Region.US_EAST_1)

.credentialsProvider(ProfileCredentialsProvider.create())
    .build();

String payload = new JSONObject()
    .put("prompt", enclosedPrompt)
    .put("max_tokens_to_sample", 200)
    .put("temperature", 0.5)
    .put("stop_sequences", List.of("\n\nHuman:"))
    .toString();

InvokeModelRequest request = InvokeModelRequest.builder()
    .body(SdkBytes.fromUtf8String(payload))
    .modelId(claudeModelId)
    .contentType("application/json")
    .accept("application/json")
    .build();

InvokeModelResponse response = client.invokeModel(request);

JSONObject responseBody = new
JSONObject(response.body().asUtf8String());

String generatedText = responseBody.getString("completion");


return generatedText;
}
```

- Untuk detail API, lihat [InvokeModel](#) di Referensi AWS SDK for Java 2.x API.

## Pembuatan teks dengan Anthropic Claude 2 dengan aliran respons

Contoh kode berikut menunjukkan cara memanggil model Anthropic Claude 2 di Amazon Bedrock untuk pembuatan teks dengan aliran respons.

## SDK for Java 2.x

 Note

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

Gunakan model Anthropic Claude 2 dan proses aliran respons.

```
/**
 * Invokes the Anthropic Claude 2 model and processes the response stream.
 *
 * @param prompt The prompt for Claude to complete.
 * @param silent Suppress console output of the individual response stream
 *              chunks.
 * @return The generated response.
 */
public static String invokeClaude(String prompt, boolean silent) {

    BedrockRuntimeAsyncClient client =
BedrockRuntimeAsyncClient.builder()
        .region(Region.US_EAST_1)

.credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    var finalCompletion = new AtomicReference<>("");

    var payload = new JSONObject()
        .put("prompt", "Human: " + prompt + " Assistant:")
        .put("temperature", 0.8)
        .put("max_tokens_to_sample", 300)
        .toString();

    var request = InvokeModelWithResponseStreamRequest.builder()
        .body(SdkBytes.fromUtf8String(payload))
        .modelId("anthropic.claude-v2")
        .contentType("application/json")
        .accept("application/json")
        .build();
```



```
        var visitor =
        InvokeModelWithResponseStreamResponseHandler.Visitor.builder()
            .onChunk(chunk -> {
                var json = new
                JSONObject(chunk.bytes().asUtf8String());
                var completion =
                json.getString("completion");
                finalCompletion.set(finalCompletion.get() +
                completion);
                if (!silent) {
                    System.out.print(completion);
                }
            })
            .build();

        var handler = InvokeModelWithResponseStreamResponseHandler.builder()
            .onEventStream(stream -> stream.subscribe(event ->
            event.accept(visitor)))
            .onComplete(() -> {
            })
            .onError(e -> System.out.println("\n\nError: " +
            e.getMessage()))
            .build();

        client.invokeModelWithResponseStream(request, handler).join();

        return finalCompletion.get();
    }
}
```

- Untuk detail API, lihat [InvokeModelWithResponseStream](#) di Referensi AWS SDK for Java 2.x API.

## Pembuatan teks dengan Meta Llama 2 Chat

Contoh kode berikut menunjukkan cara memanggil model Obrolan Meta Llama 2 di Amazon Bedrock untuk pembuatan teks.

## SDK for Java 2.x

 Note

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

Secara asinkron memanggil model dasar Meta Llama 2 Chat untuk menghasilkan teks.

```
/**
 * Asynchronously invokes the Meta Llama 2 Chat model to run an inference based
 * on the provided input.
 *
 * @param prompt The prompt that you want Llama 2 to complete.
 * @return The inference response generated by the model.
 */
public static String invokeLlama2(String prompt) {
    /**
     * The different model providers have individual request and response
    formats.
     * For the format, ranges, and default values for Meta Llama 2 Chat, refer
    to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
    meta.
     * html
     */

    String llama2ModelId = "meta.llama2-13b-chat-v1";

    BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
        .region(Region.US_EAST_1)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    String payload = new JSONObject()
        .put("prompt", prompt)
        .put("max_gen_len", 512)
        .put("temperature", 0.5)
        .put("top_p", 0.9)
        .toString();

    InvokeModelRequest request = InvokeModelRequest.builder()
```

```

        .body(SdkBytes.fromUtf8String(payload))
        .modelId(llama2ModelId)
        .contentType("application/json")
        .accept("application/json")
        .build();

    CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                System.out.println("Model invocation failed: " + exception);
            }
        });

    String generatedText = "";
    try {
        InvokeModelResponse response = completableFuture.get();
        JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
        generatedText = responseBody.getString("generation");

    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
    } catch (ExecutionException e) {
        System.err.println(e.getMessage());
    }

    return generatedText;
}

```

Panggil model dasar Meta Llama 2 Chat untuk menghasilkan teks.

```

/**
 * Invokes the Meta Llama 2 Chat model to run an inference based on the
provided
 * input.
 *
 * @param prompt The prompt for Llama 2 to complete.
 * @return The generated response.
 */
public static String invokeLlama2(String prompt) {

```

```
        /*
         * The different model providers have individual request and
         response formats.
         * For the format, ranges, and default values for Meta Llama 2 Chat,
         refer to:
         * https://docs.aws.amazon.com/bedrock/latest/userguide/model-
parameters-meta.
         * html
         */

        String llama2ModelId = "meta.llama2-13b-chat-v1";

        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
            .region(Region.US_EAST_1)

.credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        String payload = new JSONObject()
            .put("prompt", prompt)
            .put("max_gen_len", 512)
            .put("temperature", 0.5)
            .put("top_p", 0.9)
            .toString();

        InvokeModelRequest request = InvokeModelRequest.builder()
            .body(SdkBytes.fromUtf8String(payload))
            .modelId(llama2ModelId)
            .contentType("application/json")
            .accept("application/json")
            .build();

        InvokeModelResponse response = client.invokeModel(request);

        JSONObject responseBody = new
JSONObject(response.body().asUtf8String());

        String generatedText = responseBody.getString("generation");

        return generatedText;
    }
}
```

- Untuk detail API, lihat [InvokeModel](#) di Referensi AWS SDK for Java 2.x API.

## Pembuatan teks dengan Mistral 7B

Contoh kode berikut menunjukkan cara memanggil model model Mistral 7B di Amazon Bedrock untuk pembuatan teks.

SDK for Java 2.x

### Note

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

Secara asinkron memanggil model dasar Mistral 7B untuk menghasilkan teks.

```
/**
 * Asynchronously invokes the Mistral 7B model to run an inference based on the
 * provided input.
 *
 * @param prompt The prompt for Mistral to complete.
 * @return The generated response.
 */
public static List<String> invokeMistral7B(String prompt) {
    BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
        .region(Region.US_WEST_2)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    // Mistral instruct models provide optimal results when
    // embedding the prompt into the following template:
    String instruction = "<s>[INST] " + prompt + " [/INST]";

    String modelId = "mistral.mistral-7b-instruct-v0:2";

    String payload = new JSONObject()
        .put("prompt", instruction)
        .put("max_tokens", 200)
        .put("temperature", 0.5)
        .toString();

    CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request -> request
        .accept("application/json")
```

```

        .contentType("application/json")
        .body(SdkBytes.fromUtf8String(payload))
        .modelId(modelId))
    .whenComplete((response, exception) -> {
        if (exception != null) {
            System.out.println("Model invocation failed: " + exception);
        }
    });

    try {
        InvokeModelResponse response = completableFuture.get();
        JSONObject responseBody = new
        JSONObject(response.body().asUtf8String());
        JSONArray outputs = responseBody.getJSONArray("outputs");

        return IntStream.range(0, outputs.length())
            .mapToObj(i -> outputs.getJSONObject(i).getString("text"))
            .toList();
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
    } catch (ExecutionException e) {
        System.err.println(e.getMessage());
    }

    return List.of();
}

```

Panggil model pondasi Mistral 7B untuk menghasilkan teks.

```

/**
 * Invokes the Mistral 7B model to run an inference based on the provided
input.
 *
 * @param prompt The prompt for Mistral to complete.
 * @return The generated responses.
 */
public static List<String> invokeMistral7B(String prompt) {
    BedrockRuntimeClient client = BedrockRuntimeClient.builder()
        .region(Region.US_WEST_2)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();
}

```

```
// Mistral instruct models provide optimal results when
// embedding the prompt into the following template:
String instruction = "<s>[INST] " + prompt + " [/INST]";

String modelId = "mistral.mistral-7b-instruct-v0:2";

String payload = new JSONObject()
    .put("prompt", instruction)
    .put("max_tokens", 200)
    .put("temperature", 0.5)
    .toString();

InvokeModelResponse response = client.invokeModel(request -> request
    .accept("application/json")
    .contentType("application/json")
    .body(SdkBytes.fromUtf8String(payload))
    .modelId(modelId));

JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
JSONArray outputs = responseBody.getJSONArray("outputs");

return IntStream.range(0, outputs.length())
    .mapToObj(i -> outputs.getJSONObject(i).getString("text"))
    .toList();

}
```

- Untuk detail API, lihat [InvokeModel](#) di Referensi AWS SDK for Java 2.x API.

## Pembuatan teks dengan Mixtral 8x7B

Contoh kode berikut menunjukkan cara memanggil model model Mixtral 8x7B di Amazon Bedrock untuk pembuatan teks.

## SDK for Java 2.x

**Note**

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

Secara asinkron memanggil model dasar Mistral 8x7B untuk menghasilkan teks.

```
/**
 * Asynchronously invokes the Mixtral 8x7B model to run an inference based on
 the provided input.
 *
 * @param prompt The prompt for Mixtral to complete.
 * @return The generated response.
 */
public static List<String> invokeMixtral8x7B(String prompt) {
    BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
        .region(Region.US_WEST_2)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    // Mistral instruct models provide optimal results when
    // embedding the prompt into the following template:
    String instruction = "<s>[INST] " + prompt + " [/INST]";

    String modelId = "mistral.mixtral-8x7b-instruct-v0:1";

    String payload = new JSONObject()
        .put("prompt", instruction)
        .put("max_tokens", 200)
        .put("temperature", 0.5)
        .toString();

    CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request -> request
        .accept("application/json")
        .contentType("application/json")
        .body(SdkBytes.fromUtf8String(payload))
        .modelId(modelId))
        .whenComplete((response, exception) -> {
            if (exception != null) {
```



```

        System.out.println("Model invocation failed: " + exception);
    }
    });

    try {
        InvokeModelResponse response = completableFuture.get();
        JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
        JSONArray outputs = responseBody.getJSONArray("outputs");

        return IntStream.range(0, outputs.length())
            .mapToObj(i -> outputs.getJSONObject(i).getString("text"))
            .toList();
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
    } catch (ExecutionException e) {
        System.err.println(e.getMessage());
    }

    return List.of();
}

```

Panggil model dasar Mixtral 8x7B untuk menghasilkan teks.

```

public static List<String> invokeMixtral8x7B(String prompt) {
    BedrockRuntimeClient client = BedrockRuntimeClient.builder()
        .region(Region.US_WEST_2)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    // Mistral instruct models provide optimal results when
    // embedding the prompt into the following template:
    String instruction = "<s>[INST] " + prompt + " [/INST]";

    String modelId = "mistral.mixtral-8x7b-instruct-v0:1";

    String payload = new JSONObject()
        .put("prompt", instruction)
        .put("max_tokens", 200)
        .put("temperature", 0.5)
        .toString();
}

```

```
InvokeModelResponse response = client.invokeModel(request -> request
    .accept("application/json")
    .contentType("application/json")
    .body(SdkBytes.fromUtf8String(payload))
    .modelId(modelId));

JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
JSONArray outputs = responseBody.getJSONArray("outputs");

return IntStream.range(0, outputs.length())
    .mapToObj(i -> outputs.getJSONObject(i).getString("text"))
    .toList();
}
```

- Untuk detail API, lihat [InvokeModel](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

Buat aplikasi taman bermain untuk berinteraksi dengan model yayasan Amazon Bedrock

Contoh kode berikut menunjukkan cara membuat taman bermain untuk berinteraksi dengan model dasar Amazon Bedrock melalui modalitas yang berbeda.

### SDK for Java 2.x

Java Foundation Model (FM) Playground adalah contoh aplikasi Spring Boot yang menampilkan cara menggunakan Amazon Bedrock dengan Java. Contoh ini menunjukkan bagaimana pengembang Java dapat menggunakan Amazon Bedrock untuk membangun aplikasi berkemampuan AI generatif. Anda dapat menguji dan berinteraksi dengan model yayasan Amazon Bedrock dengan menggunakan tiga taman bermain berikut:

- Taman bermain teks.
- Taman bermain obrolan.
- Taman bermain gambar.

Contoh ini juga mencantumkan dan menampilkan model pondasi yang dapat Anda akses, bersama dengan karakteristiknya. Untuk kode sumber dan petunjuk penerapan, lihat proyek di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Waktu Jalan Amazon Bedrock

## CloudFront contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with CloudFront.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

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

### Tindakan

Buat distribusi

Contoh kode berikut menunjukkan cara membuat CloudFront distribusi.

SDK for Java 2.x

#### Note

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

Contoh berikut menggunakan bucket Amazon Simple Storage Service (Amazon S3) sebagai sumber konten.

Setelah membuat distribusi, kode membuat [CloudFrontWaiter](#) untuk menunggu sampai distribusi diterapkan sebelum mengembalikan distribusi.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.ItemSelection;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateDistribution {

    private static final Logger logger =
        LoggerFactory.getLogger(CreateDistribution.class);

    public static Distribution createDistribution(CloudFrontClient
        cloudFrontClient, S3Client s3Client,
        final String bucketName, final String keyGroupId, final
        String originAccessControlId) {

        final String region = s3Client.headBucket(b ->
        b.bucket(bucketName)).sdkHttpResponse().headers()
            .get("x-amz-bucket-region").get(0);
        final String originDomain = bucketName + ".s3." + region +
        ".amazonaws.com";
        String originId = originDomain; // Use the originDomain value for
        the originId.

        // The service API requires some deprecated methods, such as
        // DefaultCacheBehavior.Builder#minTTL and #forwardedValue.
        CreateDistributionResponse createDistResponse =
        cloudFrontClient.createDistribution(builder -> builder
            .distributionConfig(b1 -> b1
                .origins(b2 -> b2
```

```

        .quantity(1)
        .items(b3 -> b3

.domainName(originDomain)

.id(originId)

.s3OriginConfig(builder4 -> builder4
    .originAccessIdentity(
        ""))

.originAccessControlId(
    originAccessControlId)))
        .defaultCacheBehavior(b2 -> b2

.viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)
        .minTTL(200L)
        .forwardedValues(b5
-> b5

.cookies(cp -> cp
    .forward(ItemSelection.NONE))

.queryString(true))
        .trustedKeyGroups(b3
-> b3

.quantity(1)

.items(keyGroupId)

.enabled(true))
        .allowedMethods(b4 -
> b4

.quantity(2)

.items(Method.HEAD, Method.GET)

```

```
.cachedMethods(b5 -> b5
    .quantity(2)
    .items(Method.HEAD,
            Method.GET))))
    .cacheBehaviors(b -> b
        .quantity(1)
        .items(b2 -> b2

.pathPattern("/index.html")

.viewerProtocolPolicy(
    ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)

.trustedKeyGroups(b3 -> b3
    .quantity(1)
    .items(keyGroupId)
    .enabled(true))

.minTTL(200L)

.forwardedValues(b4 -> b4
    .cookies(cp -> cp
        .forward(ItemSelection.NONE))
    .queryString(true))

.allowedMethods(b5 -> b5.quantity(2)
    .items(Method.HEAD,
            Method.GET)
```

```

        .cachedMethods(b6 -> b6

                .quantity(2)

                .items(Method.HEAD,

                        Method.GET))))))
                .enabled(true)
                .comment("Distribution built with
java")

        .callerReference(Instant.now().toString()));

        final Distribution distribution = createDistResponse.distribution();
        logger.info("Distribution created. DomainName: [{}] Id: [{}]",
distribution.domainName(),
                distribution.id());
        logger.info("Waiting for distribution to be deployed ...");
        try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                ResponseOrException<GetDistributionResponse>
responseOrException = cfWaiter

                        .waitUntilDistributionDeployed(builder ->
builder.id(distribution.id()))

                                .matched();
                responseOrException.response()
                                .orElseThrow(() -> new
RuntimeException("Distribution not created"));
                logger.info("Distribution deployed. DomainName: [{}] Id:
[{}]", distribution.domainName(),
                                distribution.id());
        }
        return distribution;
}
}

```

- Untuk detail API, lihat [CreateDistribution](#) di Referensi AWS SDK for Java 2.x API.

## Buat fungsi

Contoh kode berikut menunjukkan cara membuat CloudFront fungsi Amazon.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionRequest;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionResponse;
import software.amazon.awssdk.services.cloudfront.model.FunctionConfig;
import software.amazon.awssdk.services.cloudfront.model.FunctionRuntime;
import java.io.InputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateFunction {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <functionName> <filePath>

            Where:
                functionName - The name of the function to create.\s
                filePath - The path to a file that contains the application
            logic for the function.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
        }
    }
}
```



```
        System.exit(1);
    }

    String functionName = args[0];
    String filePath = args[1];
    CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
        .region(Region.AWS_GLOBAL)
        .build();

    String funArn = createNewFunction(cloudFrontClient, functionName, filePath);
    System.out.println("The function ARN is " + funArn);
    cloudFrontClient.close();
}

public static String createNewFunction(CloudFrontClient cloudFrontClient, String
functionName, String filePath) {
    try {
        InputStream fileIs =
CreateFunction.class.getClassLoader().getResourceAsStream(filePath);
        SdkBytes functionCode = SdkBytes.fromInputStream(fileIs);

        FunctionConfig config = FunctionConfig.builder()
            .comment("Created by using the CloudFront Java API")
            .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .name(functionName)
            .functionCode(functionCode)
            .functionConfig(config)
            .build();

        CreateFunctionResponse response =
cloudFrontClient.createFunction(functionRequest);
        return response.functionSummary().functionMetadata().functionARN();

    } catch (CloudFrontException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Untuk detail API, lihat [CreateFunction](#) di Referensi AWS SDK for Java 2.x API.

## Buat grup kunci

Contoh kode berikut menunjukkan cara membuat grup kunci yang dapat Anda gunakan dengan URL yang ditandatangani dan cookie yang ditandatangani.

## SDK for Java 2.x

### Note

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

Grup kunci memerlukan setidaknya satu kunci publik yang digunakan untuk memverifikasi URL atau cookie yang ditandatangani.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;

import java.util.UUID;

public class CreateKeyGroup {
    private static final Logger logger =
        LoggerFactory.getLogger(CreateKeyGroup.class);

    public static String createKeyGroup(CloudFrontClient cloudFrontClient, String
publicKeyId) {
        String keyGroupId = cloudFrontClient.createKeyGroup(b -> b.keyGroupConfig(c
-> c
            .items(publicKeyId)
            .name("JavaKeyGroup" + UUID.randomUUID()))
            .keyGroup().id());
        logger.info("KeyGroup created with ID: [{}]", keyGroupId);
        return keyGroupId;
    }
}
```

- Untuk detail API, lihat [CreateKeyGroup](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus sebuah distribusi

Contoh kode berikut menunjukkan cara menghapus CloudFront distribusi.

### SDK for Java 2.x

#### Note

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

Contoh kode berikut memperbarui distribusi ke dinonaktifkan, menggunakan pelayan yang menunggu perubahan diterapkan, lalu menghapus distribusi.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;

public class DeleteDistribution {
    private static final Logger logger =
        LoggerFactory.getLogger(DeleteDistribution.class);

    public static void deleteDistribution(final CloudFrontClient
        cloudFrontClient, final String distributionId) {
        // First, disable the distribution by updating it.
        GetDistributionResponse response =
            cloudFrontClient.getDistribution(b -> b
                .id(distributionId));
        String etag = response.eTag();
        DistributionConfig distConfig =
            response.distribution().distributionConfig();
```

```

        cloudFrontClient.updateDistribution(builder -> builder
            .id(distributionId)
            .distributionConfig(builder1 -> builder1

                .cacheBehaviors(distConfig.cacheBehaviors())

                .defaultCacheBehavior(distConfig.defaultCacheBehavior()
                    .enabled(false)
                    .origins(distConfig.origins())
                    .comment(distConfig.comment()))

                .callerReference(distConfig.callerReference())

                .defaultCacheBehavior(distConfig.defaultCacheBehavior()
                    .priceClass(distConfig.priceClass())
                    .aliases(distConfig.aliases())
                    .logging(distConfig.logging()))

                .defaultRootObject(distConfig.defaultRootObject())

                .customErrorResponses(distConfig.customErrorResponses())

                .httpVersion(distConfig.httpVersion())

                .isIPV6Enabled(distConfig.isIPV6Enabled())

                .restrictions(distConfig.restrictions())

                .viewerCertificate(distConfig.viewerCertificate()
                    .webACLId(distConfig.webACLId()))

                .originGroups(distConfig.originGroups())
                    .ifMatch(etag));

        logger.info("Distribution [{}] is DISABLED, waiting for deployment
before deleting ...",
            distributionId);
        GetDistributionResponse distributionResponse;
        try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse>
responseOrException = cfWaiter
                .waitUntilDistributionDeployed(builder ->
builder.id(distributionId)).matched();

```

```

        distributionResponse = responseOrException.response()
            .orElseThrow(() -> new
RuntimeException("Could not disable distribution"));
    }

    DeleteDistributionResponse deleteDistributionResponse =
cloudFrontClient
        .deleteDistribution(builder -> builder
            .id(distributionId)

.ifMatch(distributionResponse.eTag()));
    if (deleteDistributionResponse.sdkHttpResponse().isSuccessful()) {
        logger.info("Distribution [{}] DELETED", distributionId);
    }
}
}
}

```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [DeleteDistribution](#)
  - [UpdateDistribution](#)

## Hapus sumber penandatanganan

Contoh kode berikut menunjukkan cara menghapus sumber daya yang digunakan untuk mendapatkan akses ke konten terbatas di bucket Amazon Simple Storage Service (Amazon S3).

### SDK for Java 2.x

#### Note

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

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteKeyGroupResponse;
import
software.amazon.awssdk.services.cloudfront.model.DeleteOriginAccessControlResponse;

```

```
import software.amazon.awssdk.services.cloudfront.model.DeletePublicKeyResponse;
import software.amazon.awssdk.services.cloudfront.model.GetKeyGroupResponse;
import
    software.amazon.awssdk.services.cloudfront.model.GetOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.GetPublicKeyResponse;

public class DeleteSigningResources {
    private static final Logger logger =
        LoggerFactory.getLogger(DeleteSigningResources.class);

    public static void deleteOriginAccessControl(final CloudFrontClient
cloudFrontClient,
        final String originAccessControlId) {
        GetOriginAccessControlResponse getResponse = cloudFrontClient
            .getOriginAccessControl(b -> b.id(originAccessControlId));
        DeleteOriginAccessControlResponse deleteResponse =
cloudFrontClient.deleteOriginAccessControl(builder -> builder
            .id(originAccessControlId)
            .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Origin Access Control [{}]",
originAccessControlId);
        }
    }

    public static void deleteKeyGroup(final CloudFrontClient cloudFrontClient, final
String keyGroupId) {

        GetKeyGroupResponse getResponse = cloudFrontClient.getKeyGroup(b ->
b.id(keyGroupId));
        DeleteKeyGroupResponse deleteResponse =
cloudFrontClient.deleteKeyGroup(builder -> builder
            .id(keyGroupId)
            .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Key Group [{}]", keyGroupId);
        }
    }

    public static void deletePublicKey(final CloudFrontClient cloudFrontClient,
final String publicKeyId) {
        GetPublicKeyResponse getResponse = cloudFrontClient.getPublicKey(b ->
b.id(publicKeyId));
    }
}
```

```

DeletePublicKeyResponse deleteResponse =
cloudFrontClient.deletePublicKey(builder -> builder
    .id(publicKeyId)
    .ifMatch(getResponse.eTag()));

    if (deleteResponse.sdkHttpResponse().isSuccessful()) {
        logger.info("Successfully deleted Public Key [{}]", publicKeyId);
    }
}
}

```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [DeleteKeyGroup](#)
  - [DeleteOriginAccessControl](#)
  - [DeletePublicKey](#)

## Perbarui distribusi

Contoh kode berikut menunjukkan cara memperbarui CloudFront distribusi Amazon.

## SDK for Java 2.x

### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.UpdateDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;

```

```
/**
```

```
 * Before running this Java V2 code example, set up your development
```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ModifyDistribution {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <id>\s

            Where:
                id - the id value of the distribution.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String id = args[0];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
            .region(Region.AWS_GLOBAL)
            .build();

        modDistribution(cloudFrontClient, id);
        cloudFrontClient.close();
    }

    public static void modDistribution(CloudFrontClient cloudFrontClient, String
idVal) {
        try {
            // Get the Distribution to modify.
            GetDistributionRequest disRequest = GetDistributionRequest.builder()
                .id(idVal)
                .build();

            GetDistributionResponse response =
cloudFrontClient.getDistribution(disRequest);
            Distribution disObject = response.distribution();
            DistributionConfig config = disObject.distributionConfig();
```



```

// Create a new DistributionConfig object and add new values to comment
and
// aliases
DistributionConfig config1 = DistributionConfig.builder()
    .aliases(config.aliases()) // You can pass in new values here
    .comment("New Comment")
    .cacheBehaviors(config.cacheBehaviors())
    .priceClass(config.priceClass())
    .defaultCacheBehavior(config.defaultCacheBehavior())
    .enabled(config.enabled())
    .callerReference(config.callerReference())
    .logging(config.logging())
    .originGroups(config.originGroups())
    .origins(config.origins())
    .restrictions(config.restrictions())
    .defaultRootObject(config.defaultRootObject())
    .webACLId(config.webACLId())
    .httpVersion(config.httpVersion())
    .viewerCertificate(config.viewerCertificate())
    .customErrorResponses(config.customErrorResponses())
    .build();

UpdateDistributionRequest updateDistributionRequest =
UpdateDistributionRequest.builder()
    .distributionConfig(config1)
    .id(disObject.id())
    .ifMatch(response.eTag())
    .build();

cloudFrontClient.updateDistribution(updateDistributionRequest);

} catch (CloudFrontException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
}

```

- Untuk detail API, lihat [UpdateDistribution](#) di Referensi AWS SDK for Java 2.x API.

## Unggah kunci publik

Contoh kode berikut menunjukkan cara mengunggah kunci publik.

### SDK for Java 2.x

#### Note

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

Contoh kode berikut dibaca dalam kunci publik dan mengunggahnya ke Amazon CloudFront.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreatePublicKeyResponse;
import software.amazon.awssdk.utils.IoUtils;

import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

public class CreatePublicKey {
    private static final Logger logger =
        LoggerFactory.getLogger(CreatePublicKey.class);

    public static String createPublicKey(CloudFrontClient cloudFrontClient, String
        publicKeyFileName) {
        try (InputStream is =
            CreatePublicKey.class.getClassLoader().getResourceAsStream(publicKeyFileName)) {
            String publicKeyString = IoUtils.toUtf8String(is);
            CreatePublicKeyResponse createPublicKeyResponse = cloudFrontClient
                .createPublicKey(b -> b.publicKeyConfig(c -> c
                    .name("JavaCreatedPublicKey" + UUID.randomUUID())
                    .encodedKey(publicKeyString)
                    .callerReference(UUID.randomUUID().toString())));
            String createdPublicKeyId = createPublicKeyResponse.publicKey().id();
            logger.info("Public key created with id: [{}]", createdPublicKeyId);
            return createdPublicKeyId;
        } catch (IOException e) {
```

```
        throw new RuntimeException(e);
    }
}
}
```

- Untuk detail API, lihat [CreatePublicKey](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

### Menandatangani URL dan cookie

Contoh kode berikut menunjukkan cara membuat URL dan cookie yang ditandatangani yang memungkinkan akses ke sumber daya terbatas.

### SDK for Java 2.x

#### Note

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

Gunakan [CannedSignerRequest](#) kelas untuk menandatangani URL atau cookie dengan kebijakan kalengan.

```
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCannedPolicyRequest {

    public static CannedSignerRequest createRequestForCannedPolicy(String
distributionDomainName,
        String fileNameToUpload,
        String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
```

```

        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
resourcePath).toString();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CannedSignerRequest.builder()
            .resourceUrl(cloudFrontUrl)
            .privateKey(path)
            .keyPairId(publicKeyId)
            .expirationDate(expirationDate)
            .build();
    }
}

```

Gunakan [CustomSignerRequest](#) kelas untuk menandatangani URL atau cookie dengan kebijakan khusus. Metode `activeDate` dan `ipRange` merupakan metode opsional.

```

import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCustomPolicyRequest {

    public static CustomSignerRequest createRequestForCustomPolicy(String
distributionDomainName,
        String fileNameToUpload,
        String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
resourcePath).toString();
        Instant expireDate = Instant.now().plus(7, ChronoUnit.DAYS);
        // URL will be accessible tomorrow using the signed URL.
        Instant activeDate = Instant.now().plus(1, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);
    }
}

```

```

        return CustomSignerRequest.builder()
            .resourceUrl(cloudFrontUrl)
            .privateKey(path)
            .keyPairId(publicKeyId)
            .expirationDate(expireDate)
            .activeDate(activeDate) // Optional.
            // .ipRange("192.168.0.1/24") // Optional.
            .build();
    }
}

```

Contoh berikut menunjukkan penggunaan [CloudFrontUtilities](#) kelas untuk menghasilkan cookie dan URL yang ditandatangani. [Lihat](#) contoh kode ini di GitHub.

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCannedPolicy;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCustomPolicy;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class SigningUtilities {
    private static final Logger logger =
        LoggerFactory.getLogger(SigningUtilities.class);
    private static final CloudFrontUtilities cloudFrontUtilities =
        CloudFrontUtilities.create();

    public static SignedUrl signUrlForCannedPolicy(CannedSignerRequest
        cannedSignerRequest) {
        SignedUrl signedUrl =
            cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static SignedUrl signUrlForCustomPolicy(CustomSignerRequest
        customSignerRequest) {
        SignedUrl signedUrl =
            cloudFrontUtilities.getSignedUrlWithCustomPolicy(customSignerRequest);
    }
}

```

```

        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static CookiesForCannedPolicy
    getCookiesForCannedPolicy(CannedSignerRequest cannedSignerRequest) {
        CookiesForCannedPolicy cookiesForCannedPolicy = cloudFrontUtilities
            .getCookiesForCannedPolicy(cannedSignerRequest);
        logger.info("Cookie EXPIRES header [{}]",
            cookiesForCannedPolicy.expiresHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
            cookiesForCannedPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
            cookiesForCannedPolicy.signatureHeaderValue());
        return cookiesForCannedPolicy;
    }

    public static CookiesForCustomPolicy
    getCookiesForCustomPolicy(CustomSignerRequest customSignerRequest) {
        CookiesForCustomPolicy cookiesForCustomPolicy = cloudFrontUtilities
            .getCookiesForCustomPolicy(customSignerRequest);
        logger.info("Cookie POLICY header [{}]",
            cookiesForCustomPolicy.policyHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
            cookiesForCustomPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
            cookiesForCustomPolicy.signatureHeaderValue());
        return cookiesForCustomPolicy;
    }
}

```

- Untuk detail API, lihat [CloudFrontUtilities](#) di Referensi AWS SDK for Java 2.x API.

## CloudWatch contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with CloudWatch.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.


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

Memulai

Halo CloudWatch

Contoh kode berikut menunjukkan cara untuk mulai menggunakan CloudWatch.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloService {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <namespace>\s

        Where:
```

```

        namespace - The namespace to filter against (for example, AWS/
EC2).\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String namespace = args[0];
    Region region = Region.US_EAST_1;
    CloudWatchClient cw = CloudWatchClient.builder()
        .region(region)
        .build();

    listMets(cw, namespace);
    cw.close();
}

public static void listMets(CloudWatchClient cw, String namespace) {
    try {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> System.out.println(" Retrieved metric is: "
+ metrics.metricName()));

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [ListMetrics](#) di Referensi AWS SDK for Java 2.x API.

## Topik



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

## Tindakan

### Membuat sebuah Dasbor

Contoh kode berikut menunjukkan cara membuat CloudWatch dasbor Amazon.

### SDK for Java 2.x

#### Note

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

```
public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();

        PutDashboardResponse response = cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully created.");
        List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
        if (messages.isEmpty()) {
            System.out.println("There are no messages in the new Dashboard");
        } else {
            for (DashboardValidationMessage message : messages) {
                System.out.println("Message is: " + message.message());
            }
        }
    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}

```

- Untuk detail API, lihat [PutDashboard](#) di Referensi AWS SDK for Java 2.x API.

## Buat alarm metrik

Contoh kode berikut menunjukkan cara membuat atau memperbarui CloudWatch alarm Amazon dan mengaitkannya dengan metrik yang ditentukan, ekspresi matematika metrik, model deteksi anomali, atau kueri Wawasan Metrik.

## SDK for Java 2.x

### Note

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

```
public static String createAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")

```

```

        .alarmName(alarmName)

        .comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
        .threshold(100.00)
        .metricName(customMetricName)
        .namespace(customMetricNamespace)
        .evaluationPeriods(1)
        .period(10)
        .statistic("Maximum")
        .datapointsToAlarm(1)
        .treatMissingData("ignore")
        .build();

        cw.putMetricAlarm(alarmRequest);
        System.out.println(alarmName + " was successfully created!");
        return alarmName;

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

```

- Untuk detail API, lihat [PutMetricAlarm](#) di Referensi AWS SDK for Java 2.x API.

## Membuat sebuah detektor anomali

Contoh kode berikut menunjukkan cara membuat detektor CloudWatch anomali Amazon.

SDK for Java 2.x

### Note

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

```

public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {

```

```

        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .stat("Maximum")
            .build();

        PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
            .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
            .build();

        cw.putAnomalyDetector(anomalyDetectorRequest);
        System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

- Untuk detail API, lihat [PutAnomalyDetector](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus alarm

Contoh kode berikut menunjukkan cara menghapus CloudWatch alarm Amazon.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteAlarm {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <alarmName>

            Where:
                alarmName - An alarm name to delete (for example, MyAlarm).
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_2;
        CloudWatchClient cw = CloudWatchClient.builder()
            .region(region)
```

```
        .build();

        deleteCWAlarm(cw, alarmName);
        cw.close();
    }

    public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
        try {
            DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
                .alarmNames(alarmName)
                .build();

            cw.deleteAlarms(request);
            System.out.printf("Successfully deleted alarm %s", alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DeleteAlarms](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus detektor anomali

Contoh kode berikut menunjukkan cara menghapus detektor CloudWatch anomali Amazon.

SDK for Java 2.x

### Note

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

```
public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
```

```
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .stat("Maximum")
            .build();

        DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
            .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
            .build();

        cw.deleteAnomalyDetector(request);
        System.out.println("Successfully deleted the Anomaly Detector.");


    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

- Untuk detail API, lihat [DeleteAnomalyDetector](#) di Referensi AWS SDK for Java 2.x API.

## Hapus dasbor

Contoh kode berikut menunjukkan cara menghapus CloudWatch dasbor Amazon.

## SDK for Java 2.x

 Note

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


```
public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
    try {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();
        cw.deleteDashboards(dashboardsRequest);
        System.out.println(dashboardName + " was successfully deleted.");
    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DeleteDashboards](#) di Referensi AWS SDK for Java 2.x API.

## Mendeskripsikan riwayat alarm

Contoh kode berikut menunjukkan cara menggambarkan riwayat CloudWatch alarm Amazon.

## SDK for Java 2.x

 Note

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

```
public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
```



```
try {
    // Read values from the JSON file.
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    String alarmName = rootNode.findValue("exampleAlarmName").asText();

    Instant start = Instant.parse(date);
    Instant endDate = Instant.now();
    DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
        .startDate(start)
        .endDate(endDate)
        .alarmName(alarmName)
        .historyItemType(HistoryItemType.ACTION)
        .build();

    DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
    List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
    if (historyItems.isEmpty()) {
        System.out.println("No alarm history data found for " + alarmName +
".");
    } else {
        for (AlarmHistoryItem item : historyItems) {
            System.out.println("History summary: " + item.historySummary());
            System.out.println("Time stamp: " + item.timestamp());
        }
    }

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Untuk detail API, lihat [DescribeAlarmHistory](#) di Referensi AWS SDK for Java 2.x API.

## Mendeskripsikan alarm

Contoh kode berikut menunjukkan cara mendeskripsikan CloudWatch alarm Amazon.

## SDK for Java 2.x

 Note

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

```
public static void describeAlarms(CloudWatchClient cw) {
    try {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);

        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
        List<MetricAlarm> alarmList = response.metricAlarms();
        for (MetricAlarm alarm : alarmList) {
            System.out.println("Alarm name: " + alarm.alarmName());
            System.out.println("Alarm description: " +
alarm.alarmDescription());
        }
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DescribeAlarms](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan alarm untuk metrik

Contoh kode berikut menunjukkan cara mendeskripsikan CloudWatch alarm Amazon untuk metrik.

## SDK for Java 2.x

 Note

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

```
public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        boolean hasAlarm = false;
        int retries = 10;

        DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        while (!hasAlarm && retries > 0) {
            DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
            hasAlarm = response.hasMetricAlarms();
            retries--;
            Thread.sleep(20000);
            System.out.println(".");
        }
        if (!hasAlarm)
            System.out.println("No Alarm state found for " + customMetricName +
" after 10 retries.");
        else
            System.out.println("Alarm state found for " + customMetricName +
".");
    }
}
```

```

    } catch (CloudWatchException | IOException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

- Untuk detail API, lihat [DescribeAlarmsForMetric](#) di Referensi AWS SDK for Java 2.x API.

## Mendeskripsikan detektor anomali

Contoh kode berikut menunjukkan cara menggambarkan detektor CloudWatch anomali Amazon.

### SDK for Java 2.x

#### Note

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

```

public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
            .maxResults(10)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
        List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
    }
}

```

```

        for (AnomalyDetector detector : anomalyDetectorList) {
            System.out.println("Metric name: " +
                detector.singleMetricAnomalyDetector().metricName());
            System.out.println("State: " + detector.stateValue());
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

- Untuk detail API, lihat [DescribeAnomalyDetectors](#) di Referensi AWS SDK for Java 2.x API.

## Menonaktifkan tindakan alarm

Contoh kode berikut menunjukkan cara menonaktifkan tindakan CloudWatch alarm Amazon.

### SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DisableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DisableAlarmActions {
    public static void main(String[] args) {

```

```
final String usage = ""

    Usage:
    <alarmName>

    Where:
    alarmName - An alarm name to disable (for example, MyAlarm).
    """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String alarmName = args[0];
Region region = Region.US_EAST_1;
CloudWatchClient cw = CloudWatchClient.builder()
    .region(region)
    .build();

disableActions(cw, alarmName);
cw.close();
}

public static void disableActions(CloudWatchClient cw, String alarmName) {
    try {
        DisableAlarmActionsRequest request =
DisableAlarmActionsRequest.builder()
        .alarmNames(alarmName)
        .build();

        cw.disableAlarmActions(request);
        System.out.printf("Successfully disabled actions on alarm %s",
alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DisableAlarmActions](#) di Referensi AWS SDK for Java 2.x API.

## Mengaktifkan tindakan alarm

Contoh kode berikut menunjukkan cara mengaktifkan tindakan CloudWatch alarm Amazon.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnableAlarmActions {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <alarmName>

            Where:
            alarmName - An alarm name to enable (for example, MyAlarm).
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarm = args[0];
        Region region = Region.US_EAST_1;
```

```

        CloudWatchClient cw = CloudWatchClient.builder()
            .region(region)
            .build();

        enableActions(cw, alarm);
        cw.close();
    }

    public static void enableActions(CloudWatchClient cw, String alarm) {
        try {
            EnableAlarmActionsRequest request = EnableAlarmActionsRequest.builder()
                .alarmNames(alarm)
                .build();

            cw.enableAlarmActions(request);
            System.out.printf("Successfully enabled actions on alarm %s", alarm);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Untuk detail API, lihat [EnableAlarmActions](#) di Referensi AWS SDK for Java 2.x API.

Ambil gambar data metrik

Contoh kode berikut menunjukkan cara mendapatkan gambar data CloudWatch metrik Amazon.

SDK for Java 2.x

#### Note

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

```

public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
    System.out.println("Getting Image data for custom metric.");
    try {

```



```

String myJSON = "{\n" +
    "  \"title\": \"Example Metric Graph\",\n" +
    "  \"view\": \"timeSeries\",\n" +
    "  \"stacked\": false,\n" +
    "  \"period\": 10,\n" +
    "  \"width\": 1400,\n" +
    "  \"height\": 600,\n" +
    "  \"metrics\": [\n" +
    "    [\n" +
    "      \"AWS/Billing\",\n" +
    "      \"EstimatedCharges\",\n" +
    "      \"Currency\",\n" +
    "      \"USD\"\n" +
    "    ]\n" +
    "  ]\n" +
    "}";

GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
    .metricWidget(myJSON)
    .build();

GetMetricWidgetImageResponse response =
cw.getMetricWidgetImage(imageRequest);
SdkBytes sdkBytes = response.metricWidgetImage();
byte[] bytes = sdkBytes.asByteArray();
File outputFile = new File(fileName);
try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
    outputStream.write(bytes);
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}


```

- Untuk detail API, lihat [GetMetricWidgetImage](#) di Referensi AWS SDK for Java 2.x API.

## Ambil data metrik

Contoh kode berikut menunjukkan cara mendapatkan data CloudWatch metrik Amazon.

## SDK for Java 2.x

 Note

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

```
public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set the date.
        Instant nowDate = Instant.now();

        long hours = 1;
        long minutes = 30;
        Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
ChronoUnit.MINUTES);

        Metric met = Metric.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        MetricStat metStat = MetricStat.builder()
            .stat("Maximum")
            .period(1)
            .metric(met)
            .build();

        MetricDataQuery dataQuery = MetricDataQuery.builder()
            .metricStat(metStat)
            .id("foo2")
            .returnData(true)
```

```
        .build();

List<MetricDataQuery> dq = new ArrayList<>();
dq.add(dataQuery);

GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
    .maxDatapoints(10)
    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
    .startTime(nowDate)
    .endTime(date2)
    .metricDataQueries(dq)
    .build();

GetMetricDataResponse response = cw.getMetricData(getMetReq);
List<MetricDataResult> data = response.metricDataResults();
for (MetricDataResult item : data) {
    System.out.println("The label is " + item.label());
    System.out.println("The status code is " +
item.statusCode().toString());
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Untuk detail API, lihat [GetMetricData](#) di Referensi AWS SDK for Java 2.x API.

## Ambil statistik metrik

Contoh kode berikut menunjukkan cara mendapatkan statistik CloudWatch metrik Amazon.

## SDK for Java 2.x

### Note

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

```
public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
    String metricOption, String date, Dimension myDimension) {
    try {
        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
        .endTime(endDate)
        .startTime(start)
        .dimensions(myDimension)
        .metricName(metVal)
        .namespace(nameSpace)
        .period(86400)
        .statistics(Statistic.fromValue(metricOption))
        .build();

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
                System.out
                    .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
            }
        } else {
            System.out.println("The returned data list is empty");
        }

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [GetMetricStatistics](#) di Referensi AWS SDK for Java 2.x API.

## Membuat daftar dasbor

Contoh kode berikut menunjukkan cara membuat daftar CloudWatch dasbor Amazon.

SDK for Java 2.x

### Note

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

```
public static void listDashboards(CloudWatchClient cw) {
    try {
        ListDashboardsIterable listRes = cw.listDashboardsPaginator();
        listRes.stream()
            .flatMap(r -> r.dashboardEntries().stream())
            .forEach(entry -> {
                System.out.println("Dashboard name is: " +
entry.dashboardName());
                System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
            });

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListDashboards](#) di Referensi AWS SDK for Java 2.x API.

## Membuat daftar metrik

Contoh kode berikut menunjukkan cara membuat daftar metadata untuk metrik Amazon CloudWatch . Untuk mendapatkan data untuk metrik, gunakan `GetMetricStatistics` tindakan `GetMetricData` atau.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListMetrics {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <namespace>\s

                Where:
                namespace - The namespace to filter against (for example, AWS/
                EC2).\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String namespace = args[0];
        Region region = Region.US_EAST_1;
```

```
CloudWatchClient cw = CloudWatchClient.builder()
    .region(region)
    .build();

listMets(cw, namespace);
cw.close();
}

public static void listMets(CloudWatchClient cw, String namespace) {
    boolean done = false;
    String nextToken = null;

    try {
        while (!done) {

            ListMetricsResponse response;
            if (nextToken == null) {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

                response = cw.listMetrics(request);
            } else {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .nextToken(nextToken)
                    .build();

                response = cw.listMetrics(request);
            }

            for (Metric metric : response.metrics()) {
                System.out.printf("Retrieved metric %s", metric.metricName());
                System.out.println();
            }

            if (response.nextToken() == null) {
                done = true;
            } else {
                nextToken = response.nextToken();
            }
        }
    } catch (CloudWatchException e) {
```

```

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Untuk detail API, lihat [ListMetrics](#) di Referensi AWS SDK for Java 2.x API.

Masukkan data ke dalam metrik

Contoh kode berikut menunjukkan cara mempublikasikan titik data metrik ke Amazon CloudWatch.

SDK for Java 2.x

#### Note

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

```

public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1001.00)
            .timestamp(instant)

```



```

        .build();

        MetricDatum datum2 = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1002.00)
            .timestamp(instant)
            .build();

        List<MetricDatum> metricDataList = new ArrayList<>();
        metricDataList.add(datum);
        metricDataList.add(datum2);

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace(customMetricNamespace)
            .metricData(metricDataList)
            .build();

        cw.putMetricData(request);
        System.out.println("Added metric values for for metric " +
            customMetricName);

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

- Untuk detail API, lihat [PutMetricData](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

Mulai metrik, dasbor, dan alarm CloudWatch

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- 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 for Java 2.x

### Note

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

```
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.AlarmHistoryItem;
import software.amazon.awssdk.services.cloudwatch.model.AlarmType;
import software.amazon.awssdk.services.cloudwatch.model.AnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import software.amazon.awssdk.services.cloudwatch.model.DashboardValidationMessage;
import software.amazon.awssdk.services.cloudwatch.model.Datapoint;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryResponse;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricResponse;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsRequest;
```

```
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataResponse;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageResponse;
import software.amazon.awssdk.services.cloudwatch.model.HistoryItemType;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataQuery;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataResult;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.MetricStat;
import software.amazon.awssdk.services.cloudwatch.model.PutAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.ScanBy;
import software.amazon.awssdk.services.cloudwatch.model.SingleMetricAnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import software.amazon.awssdk.services.cloudwatch.paginators.ListDashboardsIterable;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;
```

```
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To enable billing metrics and statistics for this example, make sure billing
 * alerts are enabled for your account:
 * https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/monitor\_estimated\_charges\_with\_cloudwatch.html#turning\_on\_billing\_metrics
 *
 * This Java code example performs the following tasks:
 *
 * 1. List available namespaces from Amazon CloudWatch.
 * 2. List available metrics within the selected Namespace.
 * 3. Get statistics for the selected metric over the last day.
 * 4. Get CloudWatch estimated billing for the last week.
 * 5. Create a new CloudWatch dashboard with metrics.
 * 6. List dashboards using a paginator.
 * 7. Create a new custom metric by adding data for it.
 * 8. Add the custom metric to the dashboard.
 * 9. Create an alarm for the custom metric.
 * 10. Describe current alarms.
 * 11. Get current data for the new custom metric.
 * 12. Push data into the custom metric to trigger the alarm.
 * 13. Check the alarm state using the action DescribeAlarmsForMetric.
 * 14. Get alarm history for the new alarm.
 * 15. Add an anomaly detector for the custom metric.
 * 16. Describe current anomaly detectors.
 * 17. Get a metric image for the custom metric.
 * 18. Clean up the Amazon CloudWatch resources.
 */
public class CloudWatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = ""

                Usage:
```

```

        <myDate> <costDateWeek> <dashboardName> <dashboardJson>
<dashboardAdd> <settings> <metricImage> \s

```

Where:

myDate - The start date to use to get metric statistics. (For example, 2023-01-11T18:35:24.00Z.)\s

costDateWeek - The start date to use to get AWS/Billinget statistics. (For example, 2023-01-11T18:35:24.00Z.)\s

dashboardName - The name of the dashboard to create.\s

dashboardJson - The location of a JSON file to use to create a dashboard. (See Readme file.)\s

dashboardAdd - The location of a JSON file to use to update a dashboard. (See Readme file.)\s

settings - The location of a JSON file from which various values are read. (See Readme file.)\s

metricImage - The location of a BMP file that is used to create a graph.\s

```

        """;

```

```

if (args.length != 7) {
    System.out.println(usage);
    System.exit(1);
}

```

```

Region region = Region.US_EAST_1;
String myDate = args[0];
String costDateWeek = args[1];
String dashboardName = args[2];
String dashboardJson = args[3];
String dashboardAdd = args[4];
String settings = args[5];
String metricImage = args[6];

```

```

Double dataPoint = Double.parseDouble("10.0");
Scanner sc = new Scanner(System.in);
CloudWatchClient cw = CloudWatchClient.builder()
    .region(region)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

```

```

System.out.println(DASHES);
System.out.println("Welcome to the Amazon CloudWatch example scenario.");
System.out.println(DASHES);

```

```
System.out.println(DASHES);
System.out.println(
    "1. List at least five available unique namespaces from Amazon
CloudWatch. Select one from the list.");
ArrayList<String> list = listNameSpaces(cw);
for (int z = 0; z < 5; z++) {
    int index = z + 1;
    System.out.println("    " + index + ". " + list.get(z));
}

String selectedNamespace = "";
String selectedMetrics = "";
int num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    selectedNamespace = list.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + selectedNamespace);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. List available metrics within the selected namespace
and select one from the list.");
ArrayList<String> metList = listMets(cw, selectedNamespace);
for (int z = 0; z < 5; z++) {
    int index = z + 1;
    System.out.println("    " + index + ". " + metList.get(z));
}
num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    selectedMetrics = metList.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + selectedMetrics);
Dimension myDimension = getSpecificMet(cw, selectedNamespace);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get statistics for the selected metric over the last
day.");
```

```
String metricOption = "";
ArrayList<String> statTypes = new ArrayList<>();
statTypes.add("SampleCount");
statTypes.add("Average");
statTypes.add("Sum");
statTypes.add("Minimum");
statTypes.add("Maximum");

for (int t = 0; t < 5; t++) {
    System.out.println("    " + (t + 1) + ". " + statTypes.get(t));
}
System.out.println("Select a metric statistic by entering a number from the
preceding list:");
num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    metricOption = statTypes.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + metricOption);
getAndDisplayMetricStatistics(cw, selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get CloudWatch estimated billing for the last
week.");
getMetricStatistics(cw, costDateWeek);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Create a new CloudWatch dashboard with metrics.");
createDashboardWithMetrics(cw, dashboardName, dashboardJson);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. List dashboards using a paginator.");
listDashboards(cw);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Create a new custom metric by adding data to it.");
createNewCustomMetric(cw, dataPoint);
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Add an additional metric to the dashboard.");
addMetricToDashboard(cw, dashboardAdd, dashboardName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Create an alarm for the custom metric.");
String alarmName = createAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Describe ten current alarms.");
describeAlarms(cw);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Get current data for new custom metric.");
getCustomMetricData(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Push data into the custom metric to trigger the
alarm.");
addMetricDataForAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
checkForMetricAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Get alarm history for the new alarm.");
getAlarmHistory(cw, settings, myDate);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Add an anomaly detector for the custom metric.");
addAnomalyDetector(cw, settings);
System.out.println(DASHES);
```



```

        System.out.println(DASHES);
        System.out.println("16. Describe current anomaly detectors.");
        describeAnomalyDetectors(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Get a metric image for the custom metric.");
        getAndOpenMetricImage(cw, metricImage);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up the Amazon CloudWatch resources.");
        deleteDashboard(cw, dashboardName);
        deleteCWAlarm(cw, alarmName);
        deleteAnomalyDetector(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon CloudWatch example scenario is complete.");
        System.out.println(DASHES);
        cw.close();
    }

    public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .stat("Maximum")
                .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                .singleMetricAnomalyDetector(singleMetricAnomalyDetector)

```

```
        .build();

        cw.deleteAnomalyDetector(request);
        System.out.println("Successfully deleted the Anomaly Detector.");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.println("Successfully deleted alarm " + alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
    try {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();
        cw.deleteDashboards(dashboardsRequest);
        System.out.println(dashboardName + " was successfully deleted.");

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
```

```

System.out.println("Getting Image data for custom metric.");
try {
    String myJSON = "{\n" +
        "  \"title\": \"Example Metric Graph\",\n" +
        "  \"view\": \"timeSeries\",\n" +
        "  \"stacked\": false,\n" +
        "  \"period\": 10,\n" +
        "  \"width\": 1400,\n" +
        "  \"height\": 600,\n" +
        "  \"metrics\": [\n" +
        "    [\n" +
        "      \"AWS/Billing\",\n" +
        "      \"EstimatedCharges\",\n" +
        "      \"Currency\",\n" +
        "      \"USD\"\n" +
        "    ]\n" +
        "  ]\n" +
        "}";

    GetMetricWidgetImageRequest imageRequest =
    GetMetricWidgetImageRequest.builder()
        .metricWidget(myJSON)
        .build();

    GetMetricWidgetImageResponse response =
    cw.getMetricWidgetImage(imageRequest);
    SdkBytes sdkBytes = response.metricWidgetImage();
    byte[] bytes = sdkBytes.asByteArray();
    File outputFile = new File(fileName);
    try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
        outputStream.write(bytes);
    }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));

```

```

        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
            .maxResults(10)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
        List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
        for (AnomalyDetector detector : anomalyDetectorList) {
            System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
            System.out.println("State: " + detector.stateValue());
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .stat("Maximum")

```

```

        .build();

        PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
        .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
        .build();

        cw.putAnomalyDetector(anomalyDetectorRequest);
        System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String alarmName = rootNode.findValue("exampleAlarmName").asText();

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();
        DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
        .startDate(start)
        .endDate(endDate)
        .alarmName(alarmName)
        .historyItemType(HistoryItemType.ACTION)
        .build();

        DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
        List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
        if (historyItems.isEmpty()) {
            System.out.println("No alarm history data found for " + alarmName +
".");
        } else {
            for (AlarmHistoryItem item : historyItems) {

```

```

        System.out.println("History summary: " + item.historySummary());
        System.out.println("Time stamp: " + item.timestamp());
    }
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        boolean hasAlarm = false;
        int retries = 10;

        DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        while (!hasAlarm && retries > 0) {
            DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
            hasAlarm = response.hasMetricAlarms();
            retries--;
            Thread.sleep(20000);
            System.out.println(".");
        }
        if (!hasAlarm)
            System.out.println("No Alarm state found for " + customMetricName +
" after 10 retries.");
        else
            System.out.println("Alarm state found for " + customMetricName +
".");
    }
}
}

```

```
    } catch (CloudWatchException | IOException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1001.00)
            .timestamp(instant)
            .build();

        MetricDatum datum2 = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1002.00)
            .timestamp(instant)
            .build();

        List<MetricDatum> metricDataList = new ArrayList<>();
        metricDataList.add(datum);
        metricDataList.add(datum2);

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace(customMetricNamespace)
            .metricData(metricDataList)
```

```
        .build();

        cw.putMetricData(request);
        System.out.println("Added metric values for for metric " +
customMetricName);

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set the date.
        Instant nowDate = Instant.now();

        long hours = 1;
        long minutes = 30;
        Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
ChronoUnit.MINUTES);

        Metric met = Metric.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        MetricStat metStat = MetricStat.builder()
            .stat("Maximum")
            .period(1)
            .metric(met)
            .build();

        MetricDataQuery dataQuery = MetricDataQuery.builder()
            .metricStat(metStat)
```



```
        .id("foo2")
        .returnData(true)
        .build();

List<MetricDataQuery> dq = new ArrayList<>();
dq.add(dataQuery);

GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
    .maxDatapoints(10)
    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
    .startTime(nowDate)
    .endTime(date2)
    .metricDataQueries(dq)
    .build();

GetMetricDataResponse response = cw.getMetricData(getMetReq);
List<MetricDataResult> data = response.metricDataResults();
for (MetricDataResult item : data) {
    System.out.println("The label is " + item.label());
    System.out.println("The status code is " +
item.statusCode().toString());
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void describeAlarms(CloudWatchClient cw) {
    try {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);

        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
        List<MetricAlarm> alarmList = response.metricAlarms();
        for (MetricAlarm alarm : alarmList) {
            System.out.println("Alarm name: " + alarm.alarmName());
        }
    }
}
```

```

        System.out.println("Alarm description: " +
alarm.alarmDescription());
    }
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String createAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

        .comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();
    }
}

```

```
        cw.putMetricAlarm(alarmRequest);
        System.out.println(alarmName + " was successfully created!");
        return alarmName;

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void addMetricToDashboard(CloudWatchClient cw, String fileName,
String dashboardName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();

        cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully updated.");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void createNewCustomMetric(CloudWatchClient cw, Double dataPoint)
{
    try {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
            .value("URLS")
            .build();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
```

```

        .value(dataPoint)
        .timestamp(instant)
        .dimensions(dimension)
        .build();

    PutMetricDataRequest request = PutMetricDataRequest.builder()
        .namespace("SITE/TRAFFIC")
        .metricData(datum)
        .build();

    cw.putMetricData(request);
    System.out.println("Added metric values for for metric PAGES_VISITED");

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void listDashboards(CloudWatchClient cw) {
    try {
        ListDashboardsIterable listRes = cw.listDashboardsPaginator();
        listRes.stream()
            .flatMap(r -> r.dashboardEntries().stream())
            .forEach(entry -> {
                System.out.println("Dashboard name is: " +
entry.dashboardName());
                System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
            });
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();
    }
}

```

```
        PutDashboardResponse response = cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully created.");
        List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
        if (messages.isEmpty()) {
            System.out.println("There are no messages in the new Dashboard");
        } else {
            for (DashboardValidationMessage message : messages) {
                System.out.println("Message is: " + message.message());
            }
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String readFileAsString(String file) throws IOException {
    return new String(Files.readAllBytes(Paths.get(file)));
}

public static void getMetricStatistics(CloudWatchClient cw, String costDateWeek)
{
    try {
        Instant start = Instant.parse(costDateWeek);
        Instant endDate = Instant.now();
        Dimension dimension = Dimension.builder()
            .name("Currency")
            .value("USD")
            .build();

        List<Dimension> dimensionList = new ArrayList<>();
        dimensionList.add(dimension);
        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .metricName("EstimatedCharges")
            .namespace("AWS/Billing")
            .dimensions(dimensionList)
            .statistics(Statistic.MAXIMUM)
            .startTime(start)
            .endTime(endDate)
            .period(86400)
```

```

        .build());

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
                System.out
                    .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
            }
        } else {
            System.out.println("The returned data list is empty");
        }

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
        String metricOption, String date, Dimension myDimension) {
    try {
        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .endTime(endDate)
            .startTime(start)
            .dimensions(myDimension)
            .metricName(metVal)
            .namespace(nameSpace)
            .period(86400)
            .statistics(Statistic.fromValue(metricOption))
            .build();

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {

```

```
        System.out
            .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
    }
    } else {
        System.out.println("The returned data list is empty");
    }

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static Dimension getSpecificMet(CloudWatchClient cw, String namespace) {
    try {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsResponse response = cw.listMetrics(request);
        List<Metric> myList = response.metrics();
        Metric metric = myList.get(0);
        return metric.dimensions().get(0);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static ArrayList<String> listMets(CloudWatchClient cw, String namespace)
{
    try {
        ArrayList<String> metList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> metList.add(metrics.metricName()));
    }
}
```

```
        return metList;

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static ArrayList<String> listNameSpaces(CloudWatchClient cw) {
    try {
        ArrayList<String> nameSpaceList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder()
            .build();

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> {
                String data = metrics.namespace();
                if (!nameSpaceList.contains(data)) {
                    nameSpaceList.add(data);
                }
            });

        return nameSpaceList;
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [DeleteAlarms](#)
  - [DeleteAnomalyDetector](#)
  - [DeleteDashboards](#)
  - [DescribeAlarmHistory](#)



- [DescribeAlarms](#)
- [DescribeAlarmsForMetric](#)
- [DescribeAnomalyDetectors](#)
- [GetMetricData](#)
- [GetMetricStatistics](#)
- [GetMetricWidgetImage](#)
- [ListMetrics](#)
- [PutAnomalyDetector](#)
- [PutDashboard](#)
- [PutMetricAlarm](#)
- [PutMetricData](#)

## CloudWatch Contoh acara menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x With CloudWatch Events.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

### Tindakan

Menambahkan target

Contoh kode berikut menunjukkan cara menambahkan target ke CloudWatch acara Amazon Events.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.Target;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutTargets {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <ruleName> <functionArn> <targetId>\s

            Where:
                ruleName - A rule name (for example, myrule).
                functionArn - An AWS Lambda function ARN (for example,
                arn:aws:lambda:us-west-2:xxxxxx047983:function:lamda1).
                targetId - A target id value.

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String functionArn = args[1];
```

```
String targetId = args[2];
CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
    .build();

putCWTargets(cwe, ruleName, functionArn, targetId);
cwe.close();
}

public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName,
String functionArn, String targetId) {
    try {
        Target target = Target.builder()
            .arn(functionArn)
            .id(targetId)
            .build();

        PutTargetsRequest request = PutTargetsRequest.builder()
            .targets(target)
            .rule(ruleName)
            .build();

        cwe.putTargets(request);
        System.out.printf(
            "Successfully created CloudWatch events target for rule %s",
            ruleName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [PutTargets](#) di Referensi AWS SDK for Java 2.x API.

## Buat aturan terjadwal

Contoh kode berikut menunjukkan cara membuat aturan terjadwal CloudWatch Acara Amazon.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutRule {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <ruleName> roleArn\s

            Where:
                ruleName - A rule name (for example, myrule).
                roleArn - A role ARN value (for example,
arn:aws:iam::xxxxxx047983:user/MyUser).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String roleArn = args[1];
```

```
CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
    .build();

putCWRule(cwe, ruleName, roleArn);
cwe.close();
}

public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
roleArn) {
    try {
        PutRuleRequest request = PutRuleRequest.builder()
            .name(ruleName)
            .roleArn(roleArn)
            .scheduleExpression("rate(5 minutes)")
            .state(RuleState.ENABLED)
            .build();

        PutRuleResponse response = cwe.putRule(request);
        System.out.printf(
            "Successfully created CloudWatch events rule %s with arn %s",
            ruleArn, response.ruleArn());

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [PutRule](#) di Referensi AWS SDK for Java 2.x API.

## Kirim acara

Contoh kode berikut menunjukkan cara mengirim CloudWatch acara Amazon Events.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutEvents {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <resourceArn>

            Where:
                resourceArn - An Amazon Resource Name (ARN) related to the
events.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String resourceArn = args[0];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
            .build();

        putCWEvents(cwe, resourceArn);
        cwe.close();
    }

    public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn) {
        try {
            final String EVENT_DETAILS = "{ \"key1\": \"value1\", \"key2\":
\"value2\" }";

```

```
PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
    .detail(EVENT_DETAILS)
    .detailType("sampleSubmitted")
    .resources(resourceArn)
    .source("aws-sdk-java-cloudwatch-example")
    .build();

PutEventsRequest request = PutEventsRequest.builder()
    .entries(requestEntry)
    .build();

cwe.putEvents(request);
System.out.println("Successfully put CloudWatch event");

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Untuk detail API, lihat [PutEvents](#) di Referensi AWS SDK for Java 2.x API.

## CloudWatch Contoh log menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x With CloudWatch Logs.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

## Tindakan

### Buat filter langganan

Contoh kode berikut menunjukkan cara membuat filter langganan Amazon CloudWatch Logs.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.PutSubscriptionFilterRequest;

/**
 * Before running this code example, you need to grant permission to CloudWatch
 * Logs the right to execute your Lambda function.
 * To perform this task, you can use this CLI command:
 *
 * aws lambda add-permission --function-name "lamda1" --statement-id "lamda1"
 * --principal "logs.us-west-2.amazonaws.com" --action "lambda:InvokeFunction"
 * --source-arn "arn:aws:logs:us-west-2:111111111111:log-group:testgroup:*"
 * --source-account "111111111111"
 *
 * Make sure you replace the function name with your function name and replace
 * '111111111111' with your account details.
 * For more information, see "Subscription Filters with AWS Lambda" in the
 * Amazon CloudWatch Logs Guide.
 *
 * Also, before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```



```
*/

public class PutSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <filter> <pattern> <logGroup> <functionArn>\s

            Where:
                filter - A filter name (for example, myfilter).
                pattern - A filter pattern (for example, ERROR).
                logGroup - A log group name (testgroup).
                functionArn - An AWS Lambda function ARN (for example,
arn:aws:lambda:us-west-2:111111111111:function:lambda1) .
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String pattern = args[1];
        String logGroup = args[2];
        String functionArn = args[3];
        Region region = Region.US_WEST_2;
        CloudWatchLogsClient cwl = CloudWatchLogsClient.builder()
            .region(region)
            .build();

        putSubFilters(cwl, filter, pattern, logGroup, functionArn);
        cwl.close();
    }

    public static void putSubFilters(CloudWatchLogsClient cwl,
        String filter,
        String pattern,
        String logGroup,
        String functionArn) {

        try {
            PutSubscriptionFilterRequest request =
PutSubscriptionFilterRequest.builder()
```

```

        .filterName(filter)
        .filterPattern(pattern)
        .logGroupName(logGroup)
        .destinationArn(functionArn)
        .build();

    cw1.putSubscriptionFilter(request);
    System.out.printf(
        "Successfully created CloudWatch logs subscription filter %s",
        filter);

    } catch (CloudWatchLogsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [PutSubscriptionFilter](#) di Referensi AWS SDK for Java 2.x API.

## Hapus filter langganan

Contoh kode berikut menunjukkan cara menghapus filter langganan Amazon CloudWatch Logs.

SDK for Java 2.x

### Note

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

```

import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DeleteSubscriptionFilterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 */

```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DeleteSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <filter> <logGroup>

            Where:
            filter - The name of the subscription filter (for example,
MyFilter).
            logGroup - The name of the log group. (for example, testgroup).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String logGroup = args[1];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
            .build();

        deleteSubFilter(logs, filter, logGroup);
        logs.close();
    }

    public static void deleteSubFilter(CloudWatchLogsClient logs, String filter,
String logGroup) {
        try {
            DeleteSubscriptionFilterRequest request =
DeleteSubscriptionFilterRequest.builder()
                .filterName(filter)
                .logGroupName(logGroup)
                .build();

            logs.deleteSubscriptionFilter(request);
            System.out.printf("Successfully deleted CloudWatch logs subscription
filter %s", filter);
        }
    }
}
```


```
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DeleteSubscriptionFilter](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan filter langganan yang ada

Contoh kode berikut menunjukkan cara mendeskripsikan filter langganan Amazon CloudWatch Logs yang ada.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersRequest;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.SubscriptionFilter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeSubscriptionFilters {
    public static void main(String[] args) {
```

```
final String usage = ""

    Usage:
    <logGroup>

    Where:
    logGroup - A log group name (for example, myloggroup).
    """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String logGroup = args[0];
CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

describeFilters(logs, logGroup);
logs.close();
}

public static void describeFilters(CloudWatchLogsClient logs, String logGroup) {
    try {
        boolean done = false;
        String newToken = null;

        while (!done) {
            DescribeSubscriptionFiltersResponse response;
            if (newToken == null) {
                DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                    .logGroupName(logGroup)
                    .limit(1).build();

                response = logs.describeSubscriptionFilters(request);
            } else {
                DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                    .nextToken(newToken)
                    .logGroupName(logGroup)
                    .limit(1).build();

```

```
        response = logs.describeSubscriptionFilters(request);
    }

    for (SubscriptionFilter filter : response.subscriptionFilters()) {
        System.out.printf("Retrieved filter with name %s, " + "pattern
%s " + "and destination arn %s",
            filter.filterName(),
            filter.filterPattern(),
            filter.destinationArn());
    }

    if (response.nextToken() == null) {
        done = true;
    } else {
        newToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.printf("Done");
}
}
```

- Untuk detail API, lihat [DescribeSubscriptionFilters](#) di Referensi AWS SDK for Java 2.x API.

## Memulai sesi Live Tail

Contoh kode berikut menunjukkan cara memulai sesi Live Tail untuk grup log/aliran log yang ada.

### SDK for Java 2.x

Sertakan file-file yang diperlukan.

```
import io.reactivex.FlowableSubscriber;
import io.reactivex.annotations.NonNull;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsAsyncClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionLogEvent;
```

```

import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionStart;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionUpdate;
import software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailRequest;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseHandler;
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseStream;

import java.util.Date;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;

```

Tangani acara dari sesi Live Tail.

```

    private static StartLiveTailResponseHandler
    getStartLiveTailResponseStreamHandler(
        AtomicReference<Subscription> subscriptionAtomicReference) {
        return StartLiveTailResponseHandler.builder()
            .onResponse(r -> System.out.println("Received initial response"))
            .onError(throwable -> {
                CloudWatchLogsException e = (CloudWatchLogsException)
                throwable.getCause();
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
            })
            .subscriber(() -> new FlowableSubscriber<>() {
                @Override
                public void onSubscribe(@NonNull Subscription s) {
                    subscriptionAtomicReference.set(s);
                    s.request(Long.MAX_VALUE);
                }

                @Override
                public void onNext(StartLiveTailResponseStream event) {
                    if (event instanceof LiveTailSessionStart) {
                        LiveTailSessionStart sessionStart = (LiveTailSessionStart)
                        event;

                        System.out.println(sessionStart);
                    } else if (event instanceof LiveTailSessionUpdate) {
                        LiveTailSessionUpdate sessionUpdate =
                        (LiveTailSessionUpdate) event;

```

```

        List<LiveTailSessionLogEvent> logEvents =
sessionUpdate.sessionResults();
        logEvents.forEach(e -> {
            long timestamp = e.timestamp();
            Date date = new Date(timestamp);
            System.out.println "[" + date + "] " + e.message());
        });
    } else {
        throw CloudWatchLogsException.builder().message("Unknown
event type").build();
    }
}

@Override
public void onError(Throwable throwable) {
    System.out.println(throwable.getMessage());
    System.exit(1);
}

@Override
public void onComplete() {
    System.out.println("Completed Streaming Session");
}
})
.build();
}

```

## Mulai sesi Live Tail.

```

CloudWatchLogsAsyncClient cloudWatchLogsAsyncClient =
    CloudWatchLogsAsyncClient.builder()
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

StartLiveTailRequest request =
    StartLiveTailRequest.builder()
        .logGroupIdentifiers(logGroupIdentifiers)
        .logStreamNames(logStreamNames)
        .logEventFilterPattern(logEventFilterPattern)
        .build();

/* Create a reference to store the subscription */

```



```
final AtomicReference<Subscription> subscriptionAtomicReference = new
AtomicReference<>(null);

cloudWatchLogsAsyncClient.startLiveTail(request,
getStartLiveTailResponseStreamHandler(subscriptionAtomicReference));
```

Hentikan sesi Live Tail setelah periode waktu berlalu.

```
/* Set a timeout for the session and cancel the subscription. This will:
 * 1). Close the stream
 * 2). Stop the Live Tail session
 */
try {
    Thread.sleep(10000);
} catch (InterruptedException e) {
    throw new RuntimeException(e);
}
if (subscriptionAtomicReference.get() != null) {
    subscriptionAtomicReference.get().cancel();
    System.out.println("Subscription to stream closed");
}
```

- Untuk detail API, lihat [StartLiveTail](#) di Referensi AWS SDK for Java 2.x API.

## Contoh Identitas Amazon Cognito menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan Identitas Cognito AWS SDK for Java 2.x With Amazon.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

## Topik

- [Tindakan](#)

## Tindakan

Buat kumpulan identitas

Contoh kode berikut menunjukkan cara membuat kumpulan identitas Amazon Cognito.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
    software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import
    software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <identityPoolName>\s

        Where:
```

```
        identityPoolName - The name to give your identity pool.
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String identityPoolName = args[0];
    CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
        .region(Region.US_EAST_1)
        .build();

    String identityPoolId = createIdPool(cognitoClient, identityPoolName);
    System.out.println("Unity pool ID " + identityPoolId);
    cognitoClient.close();
}

public static String createIdPool(CognitoIdentityClient cognitoClient, String
identityPoolName) {
    try {
        CreateIdentityPoolRequest poolRequest =
CreateIdentityPoolRequest.builder()
            .allowUnauthenticatedIdentities(false)
            .identityPoolName(identityPoolName)
            .build();

        CreateIdentityPoolResponse response =
cognitoClient.createIdentityPool(poolRequest);
        return response.identityPoolId();

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [CreateIdentityPool](#)
  - [ListIdentityPools](#)

## Menghapus kumpulan identitas

Contoh kode berikut menunjukkan cara menghapus kumpulan identitas Amazon Cognito.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
    software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteIdentityPool {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <identityPoolId>\s

            Where:
                identityPoolId - The Id value of your identity pool.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String identityPoolId = args[0];
CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

deleteIdPool(cognitoIdClient, identityPoolId);
cognitoIdClient.close();
}

public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
identityPoolId) {
    try {

        DeleteIdentityPoolRequest identityPoolRequest =
DeleteIdentityPoolRequest.builder()
            .identityPoolId(identityPoolId)
            .build();

        cognitoIdClient.deleteIdentityPool(identityPoolRequest);
        System.out.println("Done");

    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DeleteIdentityPool](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan kredensi untuk identitas

Contoh kode berikut menunjukkan cara mendapatkan kredensi untuk identitas Amazon Cognito.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
    software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityRequest;
import
    software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <identityId>\s

            Where:
                identityId - The Id of an existing identity in the format
                REGION:GUID.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```

String identityId = args[0];
CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
    .region(Region.US_EAST_1)
    .build();

getCredsForIdentity(cognitoClient, identityId);
cognitoClient.close();
}

public static void getCredsForIdentity(CognitoIdentityClient cognitoClient,
String identityId) {
    try {
        GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
GetCredentialsForIdentityRequest
        .builder()
        .identityId(identityId)
        .build();

        GetCredentialsForIdentityResponse response = cognitoClient
            .getCredentialsForIdentity(getCredentialsForIdentityRequest);
        System.out.println(
            "Identity ID " + response.identityId() + ", Access key ID " +
response.credentials().accessKeyId());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [GetCredentialsForIdentity](#) di Referensi AWS SDK for Java 2.x API.

## Daftar kumpulan identitas

Contoh kode berikut menunjukkan cara mendapatkan daftar kumpulan identitas Amazon Cognito.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
    software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
import
    software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listIdPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listIdPools(CognitoIdentityClient cognitoClient) {
        try {
            ListIdentityPoolsRequest poolsRequest =
                ListIdentityPoolsRequest.builder()
                    .maxResults(15)
                    .build();
```



```
ListIdentityPoolsResponse response =
cognitoClient.listIdentityPools(poolsRequest);
response.identityPools().forEach(pool -> {
    System.out.println("Pool ID: " + pool.identityPoolId());
    System.out.println("Pool name: " + pool.identityPoolName());
});

} catch (CognitoIdentityProviderException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [CreateIdentityPool](#)
  - [ListIdentityPools](#)

## Contoh Penyedia Identitas Amazon Cognito menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan Penyedia Identitas Amazon Cognito AWS SDK for Java 2.x dengan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

### Memulai

#### Halo Amazon Cognito

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon Cognito.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
        CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient)
    {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                .maxResults(10)

```

```
        .build();

        ListUserPoolsResponse response = cognitoClient.listUserPools(request);
        response.userPools().forEach(userpool -> {
            System.out.println("User pool " + userpool.name() + ", User ID " +
                userpool.id());
        });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListUserPools](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

Konfirmasikan pengguna

Contoh kode berikut menunjukkan cara mengonfirmasi pengguna Amazon Cognito.

SDK for Java 2.x

### Note

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

```
public static void confirmSignUp(CognitoIdentityProviderClient
    identityProviderClient, String clientId, String code,
        String userName) {
    try {
```

```

        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
            .username(userName)
            .build();

        identityProviderClient.confirmSignUp(signUpRequest);
        System.out.println(userName + " was confirmed");

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Untuk detail API, lihat [ConfirmSignUp](#) di Referensi AWS SDK for Java 2.x API.

Buat kumpulan pengguna

Contoh kode berikut menunjukkan cara membuat kumpulan pengguna Amazon Cognito.

SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;

/**
 * Before running this Java V2 code example, set up your development

```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateUserPool {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <userPoolName>\s

            Where:
                userPoolName - The name to give your user pool when it's
created.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolName = args[0];
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String id = createPool(cognitoClient, userPoolName);
        System.out.println("User pool ID: " + id);
        cognitoClient.close();
    }

    public static String createPool(CognitoIdentityProviderClient cognitoClient,
String userPoolName) {
        try {
            CreateUserPoolRequest request = CreateUserPoolRequest.builder()
                .poolName(userPoolName)
                .build();

            CreateUserPoolResponse response = cognitoClient.createUserPool(request);
            return response.userPool().id();
        }
    }
}
```

```

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}

```

- Untuk detail API, lihat [CreateUserPool](#) di Referensi AWS SDK for Java 2.x API.

## Buat klien aplikasi

Contoh kode berikut menunjukkan cara membuat aplikasi klien kumpulan pengguna Amazon Cognito.

## SDK for Java 2.x

### Note

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

```

import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientResponse;

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,

```

```
* and implement sign-up and sign-in flows.
*
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <clientName> <userPoolId>\s

            Where:
                clientName - The name for the user pool client to create.
                userPoolId - The ID for the user pool.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientName = args[0];
        String userPoolId = args[1];
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createPoolClient(cognitoClient, clientName, userPoolId);
        cognitoClient.close();
    }

    public static void createPoolClient(CognitoIdentityProviderClient cognitoClient,
String clientName,
    String userPoolId) {
        try {
            CreateUserPoolClientRequest request =
CreateUserPoolClientRequest.builder()
                .clientName(clientName)
                .userPoolId(userPoolId)
```

```

        .build();

        CreateUserPoolClientResponse response =
cognitoClient.createUserPoolClient(request);
        System.out.println("User pool " + response.userPoolClient().clientName()
+ " created. ID: "
        + response.userPoolClient().clientId());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [CreateUserPoolClient](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan token untuk mengaitkan aplikasi MFA dengan pengguna

Contoh kode berikut menunjukkan cara mendapatkan token untuk mengaitkan aplikasi MFA dengan pengguna Amazon Cognito.

SDK for Java 2.x

#### Note

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

```

public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
        .session(session)
        .build();

    AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
        .associateSoftwareToken(softwareTokenRequest);
    String secretCode = tokenResponse.secretCode();
}

```



```
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
    }
```

- Untuk detail API, lihat [AssociateSoftwareToken](#) di Referensi AWS SDK for Java 2.x API.

## Mendapatkan informasi tentang pengguna

Contoh kode berikut menunjukkan cara mendapatkan informasi tentang pengguna Amazon Cognito.

### SDK for Java 2.x

#### Note

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

```
public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
    String poolId) {
    try {
        AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
            .username(userName)
            .userPoolId(poolId)
            .build();

        AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
        System.out.println("User status " + response.userStatusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [AdminGetUser](#) di Referensi AWS SDK for Java 2.x API.

## Daftar kumpulan pengguna

Contoh kode berikut menunjukkan cara membuat daftar kumpulan pengguna Amazon Cognito.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
        CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient)
    {
```

```
    try {
        ListUserPoolsRequest request = ListUserPoolsRequest.builder()
            .maxResults(10)
            .build();

        ListUserPoolsResponse response = cognitoClient.listUserPools(request);
        response.userPools().forEach(userpool -> {
            System.out.println("User pool " + userpool.name() + ", User ID " +
                userpool.id());
        });

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ListUserPools](#) di Referensi AWS SDK for Java 2.x API.

## Daftar pengguna

Contoh kode berikut menunjukkan cara mencantumkan pengguna Amazon Cognito.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUsers {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <userPoolId>\s

            Where:
                userPoolId - The ID given to your user pool when it's created.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolId = args[0];
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
        cognitoClient.close();
    }

    public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {
        try {
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                .userPoolId(userPoolId)
                .build();
```

```

        ListUsersResponse response = cognitoClient.listUsers(usersRequest);
        response.users().forEach(user -> {
            System.out.println("User " + user.username() + " Status " +
                user.userStatus() + " Created "
                    + user.userCreateDate());
        });

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Shows how to list users by using a filter.
public static void listUsersFilter(CognitoIdentityProviderClient cognitoClient,
    String userPoolId) {

    try {
        String filter = "email = \"tblue@noserver.com\"";
        ListUsersRequest usersRequest = ListUsersRequest.builder()
            .userPoolId(userPoolId)
            .filter(filter)
            .build();

        ListUsersResponse response = cognitoClient.listUsers(usersRequest);
        response.users().forEach(user -> {
            System.out.println("User with filter applied " + user.username() + "
                Status " + user.userStatus()
                    + " Created " + user.userCreateDate());
        });

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [ListUsers](#) di Referensi AWS SDK for Java 2.x API.

## Kirim ulang kode konfirmasi

Contoh kode berikut menunjukkan cara mengirim ulang kode konfirmasi Amazon Cognito.

SDK for Java 2.x

### Note

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

```
public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
    String userName) {
    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
            .clientId(clientId)
            .username(userName)
            .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ResendConfirmationCode](#) di Referensi AWS SDK for Java 2.x API.

## Menanggapi tantangan otentikasi

Contoh kode berikut menunjukkan cara menanggapi tantangan otentikasi Amazon Cognito.

## SDK for Java 2.x

 Note

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

```
// Respond to an authentication challenge.
public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
    String userName, String clientId, String mfaCode, String session) {
    System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
    Map<String, String> challengeResponses = new HashMap<>();

    challengeResponses.put("USERNAME", userName);
    challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

    AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
        .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
        .clientId(clientId)
        .challengeResponses(challengeResponses)
        .session(session)
        .build();

    AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
        .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
    System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
        + respondToAuthChallengeResult.authenticationResult());
}
```

- Untuk detail API, lihat [AdminRespondToAuthChallenge](#) di Referensi AWS SDK for Java 2.x API.

## Mendaftar pengguna

Contoh kode berikut menunjukkan cara mendaftar pengguna dengan Amazon Cognito.

## SDK for Java 2.x

 Note

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

```
public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
    String password, String email) {
    AttributeType userAttrs = AttributeType.builder()
        .name("email")
        .value(email)
        .build();

    List<AttributeType> userAttrsList = new ArrayList<>();
    userAttrsList.add(userAttrs);
    try {
        SignUpRequest signUpRequest = SignUpRequest.builder()
            .userAttributes(userAttrsList)
            .username(userName)
            .clientId(clientId)
            .password(password)
            .build();

        identityProviderClient.signUp(signUpRequest);
        System.out.println("User has been signed up ");

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [SignUp](#) di Referensi AWS SDK for Java 2.x API.



## Mulai otentikasi dengan kredensi administrator

Contoh kode berikut menunjukkan cara memulai otentikasi dengan Amazon Cognito dan kredensial administrator.

SDK for Java 2.x

### Note

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

```
public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
             String clientId, String userName, String password, String userPoolId) {
    try {
        Map<String, String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                        .clientId(clientId)
                        .userPoolId(userPoolId)
                        .authParameters(authParameters)
                        .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                        .build();

        AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
        System.out.println("Result Challenge is : " + response.challengeName());
        return response;

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Untuk detail API, lihat [AdminInitiateAuth](#) di Referensi AWS SDK for Java 2.x API.

## Verifikasi aplikasi MFA dengan pengguna

Contoh kode berikut menunjukkan cara memverifikasi aplikasi MFA dengan pengguna Amazon Cognito.

### SDK for Java 2.x

#### Note

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

```
// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
            .userCode(code)
            .session(session)
            .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is " +
verifyResponse.statusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [VerifySoftwareToken](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

Mendaftar pengguna dengan kumpulan pengguna yang membutuhkan MFA

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- 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 for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenResponse;
import software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
```

```
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeResponse;
import software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenResponse;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito\_scenario\_user\_pool\_with\_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 * to set up TOTP (time-based one-time password). (The response is
 * "ChallengeName": "MFA_SETUP").
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
```

```

* key. This can be used with Google Authenticator.
* 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
* MFA.
* 8. Invokes the AdminInitiateAuth to sign in again. This results in being
* prompted to submit a TOTP (Response: "ChallengeName": "SOFTWARE_TOKEN_MFA").
* 9. Invokes the AdminRespondToAuthChallenge to get back a token.
*/

```

```

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws NoSuchAlgorithmException,
    InvalidKeyException {
        final String usage = ""

            Usage:
                <clientId> <poolId>

            Where:
                clientId - The app client Id value that you can get from the AWS
CDK script.
                poolId - The pool Id that you can get from the AWS CDK script.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

```

```
System.out.println("*** Enter your password");
String password = in.nextLine();

System.out.println("*** Enter your email");
String email = in.nextLine();

System.out.println("1. Signing up " + userName);
signUp(identityProviderClient, clientId, userName, password, email);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Getting " + userName + " in the user pool");
getAdminUser(identityProviderClient, userName, poolId);

System.out
    .println("*** Conformation code sent to " + userName + ". Would you
like to send a new code? (Yes/No)");
System.out.println(DASHES);

System.out.println(DASHES);
String ans = in.nextLine();

if (ans.compareTo("Yes") == 0) {
    resendConfirmationCode(identityProviderClient, clientId, userName);
    System.out.println("3. Sending a new confirmation code");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Enter confirmation code that was emailed");
String code = in.nextLine();
confirmSignUp(identityProviderClient, clientId, code, userName);
System.out.println("Rechecking the status of " + userName + " in the user
pool");
getAdminUser(identityProviderClient, userName, poolId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Invokes the initiateAuth to sign in");
AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
    poolId);
String mySession = authResponse.session();
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to generate
a TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient, mySession);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
        AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
```

```
        String userName, String clientId, String mfaCode, String session) {
    System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
    Map<String, String> challengeResponses = new HashMap<>();

    challengeResponses.put("USERNAME", userName);
    challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

    AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
        .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
        .clientId(clientId)
        .challengeResponses(challengeResponses)
        .session(session)
        .build();

    AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
        .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
    System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
        + respondToAuthChallengeResult.authenticationResult());
}

// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
            .userCode(code)
            .session(session)
            .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is " +
verifyResponse.statusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



```
public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
             String clientId, String userName, String password, String userPoolId) {
    try {
        Map<String, String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                        .clientId(clientId)
                        .userPoolId(userPoolId)
                        .authParameters(authParameters)
                        .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                        .build();

        AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
        System.out.println("Result Challenge is : " + response.challengeName());
        return response;

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}

public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
                        .session(session)
                        .build();

    AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
        .associateSoftwareToken(softwareTokenRequest);
    String secretCode = tokenResponse.secretCode();
    System.out.println("Enter this token into Google Authenticator");
    System.out.println(secretCode);
    return tokenResponse.session();
}
```

```
public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
    String userName) {
    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
            .username(userName)
            .build();

        identityProviderClient.confirmSignUp(signUpRequest);
        System.out.println(userName + " was confirmed");

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
    String userName) {
    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
            .clientId(clientId)
            .username(userName)
            .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
    String password, String email) {
    AttributeType userAttrs = AttributeType.builder()
```

```
        .name("email")
        .value(email)
        .build();

List<AttributeType> userAttrsList = new ArrayList<>();
userAttrsList.add(userAttrs);
try {
    SignUpRequest signUpRequest = SignUpRequest.builder()
        .userAttributes(userAttrsList)
        .username(userName)
        .clientId(clientId)
        .password(password)
        .build();

    identityProviderClient.signUp(signUpRequest);
    System.out.println("User has been signed up ");

} catch (CognitoIdentityProviderException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
String poolId) {
    try {
        AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
            .username(userName)
            .userPoolId(poolId)
            .build();

        AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
        System.out.println("User status " + response.userStatusAsString());

    } catch (CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [AdminGetUser](#)
  - [AdminInitiateAuth](#)
  - [AdminRespondToAuthChallenge](#)
  - [AssociateSoftwareToken](#)
  - [ConfirmDevice](#)
  - [ConfirmSignUp](#)
  - [InitiateAuth](#)
  - [ListUsers](#)
  - [ResendConfirmationCode](#)
  - [RespondToAuthChallenge](#)
  - [SignUp](#)
  - [VerifySoftwareToken](#)

## Amazon Comprehend contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan Amazon AWS SDK for Java 2.x Comprehend.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

## Tindakan

Buat pengklasifikasi dokumen

Contoh kode berikut menunjukkan cara membuat pengklasifikasi dokumen Amazon Comprehend.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
    software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierRequest;
import
    software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierResponse;
import
    software.amazon.awssdk.services.comprehend.model.DocumentClassifierInputDataConfig;

/**
 * Before running this code example, you can setup the necessary resources, such
 * as the CSV file and IAM Roles, by following this document:
 * https://aws.amazon.com/blogs/machine-learning/building-a-custom-classifier-using-
amazon-comprehend/
 *
 * Also, set up your development environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DocumentClassifierDemo {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <dataAccessRoleArn> <s3Uri> <documentClassifierName>

                Where:
```

```
        dataAccessRoleArn - The ARN value of the role used for this
operation.
        s3Uri - The Amazon S3 bucket that contains the CSV file.
        documentClassifierName - The name of the document classifier.
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String dataAccessRoleArn = args[0];
    String s3Uri = args[1];
    String documentClassifierName = args[2];

    Region region = Region.US_EAST_1;
    ComprehendClient comClient = ComprehendClient.builder()
        .region(region)
        .build();

    createDocumentClassifier(comClient, dataAccessRoleArn, s3Uri,
documentClassifierName);
    comClient.close();
}

public static void createDocumentClassifier(ComprehendClient comClient, String
dataAccessRoleArn, String s3Uri,
    String documentClassifierName) {
    try {
        DocumentClassifierInputDataConfig config =
DocumentClassifierInputDataConfig.builder()
            .s3Uri(s3Uri)
            .build();

        CreateDocumentClassifierRequest createDocumentClassifierRequest =
CreateDocumentClassifierRequest.builder()
            .documentClassifierName(documentClassifierName)
            .dataAccessRoleArn(dataAccessRoleArn)
            .languageCode("en")
            .inputDataConfig(config)
            .build();

        CreateDocumentClassifierResponse createDocumentClassifierResult =
comClient
```

```

        .createDocumentClassifier(createDocumentClassifierRequest);
        String documentClassifierArn =
createDocumentClassifierResult.documentClassifierArn();
        System.out.println("Document Classifier ARN: " + documentClassifierArn);

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Untuk detail API, lihat [CreateDocumentClassifier](#) di Referensi AWS SDK for Java 2.x API.

## Mendeteksi entitas dalam dokumen

Contoh kode berikut menunjukkan cara mendeteksi entitas dalam dokumen dengan Amazon Comprehend.

SDK for Java 2.x

### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesResponse;
import software.amazon.awssdk.services.comprehend.model.Entity;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:

```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DetectEntities {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectEntities");
        detectAllEntities(comClient, text);
        comClient.close();
    }

    public static void detectAllEntities(ComprehendClient comClient, String text) {
        try {
            DetectEntitiesRequest detectEntitiesRequest =
            DetectEntitiesRequest.builder()
                .text(text)
                .languageCode("en")
                .build();

            DetectEntitiesResponse detectEntitiesResult =
            comClient.detectEntities(detectEntitiesRequest);
            List<Entity> entList = detectEntitiesResult.entities();
            for (Entity entity : entList) {
                System.out.println("Entity text is " + entity.text());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DetectEntities](#) di Referensi AWS SDK for Java 2.x API.



## Mendeteksi frase kunci dalam dokumen

Contoh kode berikut menunjukkan cara mendeteksi frasa kunci dalam dokumen dengan Amazon Comprehend.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesResponse;
import software.amazon.awssdk.services.comprehend.model.KeyPhrase;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectKeyPhrases {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectKeyPhrases");
        detectAllKeyPhrases(comClient, text);
    }
}
```

```
        comClient.close();
    }

    public static void detectAllKeyPhrases(ComprehendClient comClient, String text)
    {
        try {
            DetectKeyPhrasesRequest detectKeyPhrasesRequest =
            DetectKeyPhrasesRequest.builder()
                .text(text)
                .languageCode("en")
                .build();

            DetectKeyPhrasesResponse detectKeyPhrasesResult =
            comClient.detectKeyPhrases(detectKeyPhrasesRequest);
            List<KeyPhrase> phraseList = detectKeyPhrasesResult.keyPhrases();
            for (KeyPhrase keyPhrase : phraseList) {
                System.out.println("Key phrase text is " + keyPhrase.text());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DetectKeyPhrases](#) di Referensi AWS SDK for Java 2.x API.

## Mendeteksi elemen sintaksis dokumen

Contoh kode berikut menunjukkan cara mendeteksi elemen sintaksis dokumen dengan Amazon Comprehend.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxResponse;
import software.amazon.awssdk.services.comprehend.model.SyntaxToken;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectSyntax {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectSyntax");
        detectAllSyntax(comClient, text);
        comClient.close();
    }

    public static void detectAllSyntax(ComprehendClient comClient, String text) {
        try {
            DetectSyntaxRequest detectSyntaxRequest = DetectSyntaxRequest.builder()
                .text(text)
                .languageCode("en")
                .build();

            DetectSyntaxResponse detectSyntaxResult =
comClient.detectSyntax(detectSyntaxRequest);
            List<SyntaxToken> syntaxTokens = detectSyntaxResult.syntaxTokens();
            for (SyntaxToken token : syntaxTokens) {
```

```

        System.out.println("Language is " + token.text());
        System.out.println("Part of speech is " +
token.partOfSpeech().tagAsString());
    }

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- Untuk detail API, lihat [DetectSyntax](#) di Referensi AWS SDK for Java 2.x API.

## Mendeteksi bahasa dominan dalam dokumen

Contoh kode berikut menunjukkan cara mendeteksi bahasa dominan dalam dokumen dengan Amazon Comprehend.

### SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
    software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageRequest;
import
    software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageResponse;
import software.amazon.awssdk.services.comprehend.model.DominantLanguage;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.

```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DetectLanguage {
    public static void main(String[] args) {
        // Specify French text - "It is raining today in Seattle".
        String text = "Il pleut aujourd'hui à Seattle";
        Region region = Region.US_EAST_1;

        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectDominantLanguage");
        detectTheDominantLanguage(comClient, text);
        comClient.close();
    }

    public static void detectTheDominantLanguage(ComprehendClient comClient, String
text) {
        try {
            DetectDominantLanguageRequest request =
DetectDominantLanguageRequest.builder()
                .text(text)
                .build();

            DetectDominantLanguageResponse resp =
comClient.detectDominantLanguage(request);
            List<DominantLanguage> allLanList = resp.languages();
            for (DominantLanguage lang : allLanList) {
                System.out.println("Language is " + lang.languageCode());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DetectDominantLanguage](#) di Referensi AWS SDK for Java 2.x API.

## Mendeteksi sentimen dokumen

Contoh kode berikut menunjukkan cara mendeteksi sentimen dokumen dengan Amazon Comprehend.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectSentiment {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectSentiment");
        detectSentiments(comClient, text);
        comClient.close();
    }
}
```

```
public static void detectSentiments(ComprehendClient comClient, String text) {
    try {
        DetectSentimentRequest detectSentimentRequest =
DetectSentimentRequest.builder()
            .text(text)
            .languageCode("en")
            .build();

        DetectSentimentResponse detectSentimentResult =
comClient.detectSentiment(detectSentimentRequest);
        System.out.println("The Neutral value is " +
detectSentimentResult.sentimentScore().neutral());

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DetectSentiment](#) di Referensi AWS SDK for Java 2.x API.

## Contoh DynamoDB menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with DynamoDB.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Memulai

## Halo DynamoDB

Contoh kode berikut ini menunjukkan cara untuk mulai menggunakan DynamoDB.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
```



```
try {
    ListTablesResponse response = null;
    if (lastName == null) {
        ListTablesRequest request = ListTablesRequest.builder().build();
        response = ddb.listTables(request);
    } else {
        ListTablesRequest request = ListTablesRequest.builder()
            .exclusiveStartTableName(lastName).build();
        response = ddb.listTables(request);
    }

    List<String> tableNames = response.tableNames();
    if (tableNames.size() > 0) {
        for (String curName : tableNames) {
            System.out.format("* %s\n", curName);
        }
    } else {
        System.out.println("No tables found!");
        System.exit(0);
    }

    lastName = response.lastEvaluatedTableName();
    if (lastName == null) {
        moreTables = false;
    }

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
System.out.println("\nDone!");
}
```

- Untuk detail API, lihat [ListTables](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

### Buat tabel

Contoh kode berikut menunjukkan cara membuat tabel DynamoDB.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTable {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <tableName> <key>
```

```

        Where:
            tableName - The Amazon DynamoDB table to create (for example,
Music3).
            key - The key for the Amazon DynamoDB table (for example,
Artist).
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String tableName = args[0];
    String key = args[1];
    System.out.println("Creating an Amazon DynamoDB table " + tableName + " with
a simple primary key: " + key);
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    String result = createTable(ddb, tableName, key);
    System.out.println("New table is " + result);
    ddb.close();
}

public static String createTable(DynamoDbClient ddb, String tableName, String
key) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    CreateTableRequest request = CreateTableRequest.builder()
        .attributeDefinitions(AttributeDefinition.builder()
            .attributeName(key)
            .attributeType(ScalarAttributeType.S)
            .build())
        .keySchema(KeySchemaElement.builder()
            .attributeName(key)
            .keyType(KeyType.HASH)
            .build())
        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(10L)
            .writeCapacityUnits(10L)
            .build())
        .tableName(tableName)

```

```
        .build();

String newTable;
try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    newTable = response.tableDescription().tableName();
    return newTable;

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
}
```

- Untuk detail API, lihat [CreateTable](#) di Referensi AWS SDK for Java 2.x API.

## Hapus tabel

Contoh kode berikut menunjukkan cara menghapus tabel DynamoDB.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
```

```
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteTable {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <tableName>

                Where:
                tableName - The Amazon DynamoDB table to delete (for example,
Music3).

                **Warning** This program will delete the table that you specify!
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        System.out.format("Deleting the Amazon DynamoDB table %s...\n", tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
    }
```

```
        .build();

    try {
        ddb.deleteTable(request);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}
}
```

- Untuk detail API, lihat [DeleteTable](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus item dalam tabel

Contoh kode berikut menunjukkan cara menghapus item dari tabel DynamoDB.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
*/
public class DeleteItem {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <key> <keyval>

            Where:
                tableName - The Amazon DynamoDB table to delete the item from
(for example, Music3).
                key - The key used in the Amazon DynamoDB table (for example,
Artist).\s
                keyval - The key value that represents the item to delete (for
example, Famous Band).
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Deleting item \"%s\" from %s\n", keyVal, tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        deleteDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }

    public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName,
String key, String keyVal) {
        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put(key, AttributeValue.builder()
            .s(keyVal)
            .build());

        DeleteItemRequest deleteReq = DeleteItemRequest.builder()
            .tableName(tableName)
```

```
        .key(keyToGet)
        .build();

    try {
        ddb.deleteItem(deleteReq);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DeleteItem](#) di Referensi AWS SDK for Java 2.x API.

## Mendapatkan batch item

Contoh kode berikut menunjukkan bagaimana untuk mendapatkan batch item DynamoDB.

### SDK for Java 2.x

#### Note

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

menunjukkan cara mendapatkan item batch menggunakan klien layanan.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
```



```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class BatchReadItems {
    public static void main(String[] args){
        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        getBatchItems(dynamoDbClient, tableName);
    }

    public static void getBatchItems(DynamoDbClient dynamoDbClient, String
tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
        key1.put("Artist", AttributeValue.builder().s("Artist1").build());

        Map<String, AttributeValue> key2 = new HashMap<>();
        key2.put("Artist", AttributeValue.builder().s("Artist2").build());

        // Construct the batchGetItem request.
        Map<String, KeysAndAttributes> requestItems = new HashMap<>();
        requestItems.put(tableName, KeysAndAttributes.builder()
            .keys(List.of(key1, key2))
            .projectionExpression("Artist, SongTitle")
            .build());

        BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
            .requestItems(requestItems)
            .build();
    }
}
```

```

    // Make the batchGetItem request.
    BatchGetItemResponse batchGetItemResponse =
dynamodbClient.batchGetItem(batchGetItemRequest);

    // Extract and print the retrieved items.
    Map<String, List<Map<String, AttributeValue>>> responses =
batchGetItemResponse.responses();
    if (responses.containsKey(tableName)) {
        List<Map<String, AttributeValue>> musicItems = responses.get(tableName);
        for (Map<String, AttributeValue> item : musicItems) {
            System.out.println("Artist: " + item.get("Artist").s() +
                ", SongTitle: " + item.get("SongTitle").s());
        }
    } else {
        System.out.println("No items retrieved.");
    }
}
}

```

menunjukkan cara mendapatkan item batch menggunakan klien layanan dan paginator.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class BatchGetItemsPaginator {

    public static void main(String[] args){
        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table (for example, Music).\s

```

```

        """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        getBatchItemsPaginator(dynamoDbClient, tableName) ;
    }

    public static void getBatchItemsPaginator(DynamoDbClient dynamoDbClient, String
tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
        key1.put("Artist", AttributeValue.builder().s("Artist1").build());

        Map<String, AttributeValue> key2 = new HashMap<>();
        key2.put("Artist", AttributeValue.builder().s("Artist2").build());

        // Construct the batchGetItem request.
        Map<String, KeysAndAttributes> requestItems = new HashMap<>();
        requestItems.put(tableName, KeysAndAttributes.builder()
            .keys(List.of(key1, key2))
            .projectionExpression("Artist, SongTitle")
            .build());

        BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
            .requestItems(requestItems)
            .build();

        // Use batchGetItemPaginator for paginated requests.
        dynamoDbClient.batchGetItemPaginator(batchGetItemRequest).stream()
            .flatMap(response -> response.responses().getOrDefault(tableName,
Collections.emptyList()).stream())
            .forEach(item -> {
                System.out.println("Artist: " + item.get("Artist").s() +
                    ", SongTitle: " + item.get("SongTitle").s());
            });
    }
}

```

- Untuk detail API, lihat [BatchGetItem](#) di Referensi AWS SDK for Java 2.x API.

## Mendapatkan item dari tabel

Contoh kode berikut menunjukkan bagaimana untuk mendapatkan item dari tabel DynamoDB.

### SDK for Java 2.x

#### Note

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

Mendapat item dari tabel dengan menggunakan DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To get an item from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, see the EnhancedGetItem example.
 */
public class GetItem {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <tableName> <key> <keyVal>

                Where:
```

```
        tableName - The Amazon DynamoDB table from which an item is
retrieved (for example, Music3).\s
        key - The key used in the Amazon DynamoDB table (for example,
Artist).\s
        keyval - The key value that represents the item to get (for
example, Famous Band).
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String tableName = args[0];
    String key = args[1];
    String keyVal = args[2];
    System.out.format("Retrieving item \"%s\" from \"%s\"\n", keyVal,
tableName);
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    getDynamoDBItem(ddb, tableName, key, keyVal);
    ddb.close();
}

public static void getDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal) {
    HashMap<String, AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal)
        .build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName(tableName)
        .build();

    try {
        // If there is no matching item, GetItem does not return any data.
        Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();
        if (returnedItem.isEmpty())
            System.out.format("No item found with the key %s!\n", key);
    }
}
```

```
        else {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");
            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
            }
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [GetItem](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan informasi tentang tabel

Contoh kode berikut menunjukkan bagaimana untuk mendapatkan informasi tentang tabel DynamoDB.

SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import
    software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeTable {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table to get information about
(for example, Music3).
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        System.out.format("Getting description for %s\n\n", tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        describeDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void describeDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DescribeTableRequest request = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            TableDescription tableInfo = ddb.describeTable(request).table();
            if (tableInfo != null) {
```

```

        System.out.format("Table name   : %s\n", tableInfo.tableName());
        System.out.format("Table ARN    : %s\n", tableInfo.tableArn());
        System.out.format("Status      : %s\n", tableInfo.tableStatus());
        System.out.format("Item count   : %d\n", tableInfo.itemCount());
        System.out.format("Size (bytes): %d\n", tableInfo.tableSizeBytes());

        ProvisionedThroughputDescription throughputInfo =
tableInfo.provisionedThroughput();
        System.out.println("Throughput");
        System.out.format("  Read Capacity : %d\n",
throughputInfo.readCapacityUnits());
        System.out.format("  Write Capacity: %d\n",
throughputInfo.writeCapacityUnits());

        List<AttributeDefinition> attributes =
tableInfo.attributeDefinitions();
        System.out.println("Attributes");
        for (AttributeDefinition a : attributes) {
            System.out.format("  %s (%s)\n", a.attributeName(),
a.attributeType());
        }
    }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("\nDone!");
}
}

```


- Untuk detail API, lihat [DescribeTable](#) di Referensi AWS SDK for Java 2.x API.

## Mencantumkan tabel

Contoh kode berikut menunjukkan bagaimana daftar tabel DynamoDB.



## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
```

```
        ListTablesRequest request = ListTablesRequest.builder().build();
        response = ddb.listTables(request);
    } else {
        ListTablesRequest request = ListTablesRequest.builder()
            .exclusiveStartTableName(lastName).build();
        response = ddb.listTables(request);
    }

    List<String> tableNames = response.tableNames();
    if (tableNames.size() > 0) {
        for (String curName : tableNames) {
            System.out.format("* %s\n", curName);
        }
    } else {
        System.out.println("No tables found!");
        System.exit(0);
    }

    lastName = response.lastEvaluatedTableName();
    if (lastName == null) {
        moreTables = false;
    }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
System.out.println("\nDone!");
}
```

- Untuk detail API, lihat [ListTables](#) di Referensi AWS SDK for Java 2.x API.

## Menempatkan item dalam tabel

Contoh kode berikut menunjukkan bagaimana menempatkan item dalam tabel DynamoDB.

## SDK for Java 2.x

 Note

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

Menempatkan item ke dalam tabel menggunakan [DynamoDbClient](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To place items into an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client. See the EnhancedPutItem example.
 */
public class PutItem {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <key> <keyVal> <albumtitle> <albumtitleval> <awards>
<awardsval> <Songtitle> <songtitleval>

            Where:
                tableName - The Amazon DynamoDB table in which an item is placed
(for example, Music3).
```

```

        key - The key used in the Amazon DynamoDB table (for example,
Artist).
        keyval - The key value that represents the item to get (for
example, Famous Band).
        albumTitle - The Album title (for example, AlbumTitle).
        AlbumTitleValue - The name of the album (for example, Songs
About Life ).
        Awards - The awards column (for example, Awards).
        AwardVal - The value of the awards (for example, 10).
        SongTitle - The song title (for example, SongTitle).
        SongTitleVal - The value of the song title (for example, Happy
Day).

        **Warning** This program will place an item that you specify into a
table!

        """;

    if (args.length != 9) {
        System.out.println(usage);
        System.exit(1);
    }

    String tableName = args[0];
    String key = args[1];
    String keyVal = args[2];
    String albumTitle = args[3];
    String albumTitleValue = args[4];
    String awards = args[5];
    String awardVal = args[6];
    String songTitle = args[7];
    String songTitleVal = args[8];

    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    putItemInTable(ddb, tableName, key, keyVal, albumTitle, albumTitleValue,
awards, awardVal, songTitle,
        songTitleVal);
    System.out.println("Done!");
    ddb.close();
}

public static void putItemInTable(DynamoDbClient ddb,

```

```

        String tableName,
        String key,
        String keyVal,
        String albumTitle,
        String albumTitleValue,
        String awards,
        String awardVal,
        String songTitle,
        String songTitleVal) {

    HashMap<String, AttributeValue> itemValues = new HashMap<>();
    itemValues.put(key, AttributeValue.builder().s(keyVal).build());
    itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
    itemValues.put(albumTitle,
AttributeValue.builder().s(albumTitleValue).build());
    itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

    PutItemRequest request = PutItemRequest.builder()
        .tableName(tableName)
        .item(itemValues)
        .build();

    try {
        PutItemResponse response = ddb.putItem(request);
        System.out.println(tableName + " was successfully updated. The request
id is "
            + response.responseMetadata().requestId());

    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.err.println("Be sure that it exists and that you've typed its
name correctly!");
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [PutItem](#) di Referensi AWS SDK for Java 2.x API.

## Mengkueri Tabel

Contoh kode berikut menunjukkan bagaimana untuk query tabel DynamoDB.

SDK for Java 2.x

### Note

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

Kueri tabel dengan menggunakan [DynamoDbClient](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To query items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client. See the EnhancedQueryRecords example.
 */
public class Query {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <tableName> <partitionKeyName> <partitionKeyVal>

                Where:
```

```

        tableName - The Amazon DynamoDB table to put the item in (for
example, Music3).
        partitionKeyName - The partition key name of the Amazon DynamoDB
table (for example, Artist).
        partitionKeyVal - The value of the partition key that should
match (for example, Famous Band).
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String tableName = args[0];
    String partitionKeyName = args[1];
    String partitionKeyVal = args[2];

    // For more information about an alias, see:
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Expressions.ExpressionAttributeNames.html
    String partitionAlias = "#a";

    System.out.format("Querying %s", tableName);
    System.out.println("");
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    int count = queryTable(ddb, tableName, partitionKeyName, partitionKeyVal,
partitionAlias);
    System.out.println("There were " + count + " record(s) returned");
    ddb.close();
}

public static int queryTable(DynamoDbClient ddb, String tableName, String
partitionKeyName, String partitionKeyVal,
    String partitionAlias) {
    // Set up an alias for the partition key name in case it's a reserved word.
    HashMap<String, String> attrNameAlias = new HashMap<String, String>();
    attrNameAlias.put(partitionAlias, partitionKeyName);

    // Set up mapping of the partition name with the value.
    HashMap<String, AttributeValue> attrValues = new HashMap<>();

```

```

        attrValues.put(":" + partitionKeyName, AttributeValue.builder()
            .s(partitionKeyVal)
            .build());

        QueryRequest queryReq = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(partitionAlias + " = :" + partitionKeyName)
            .expressionAttributeNames(attrNameAlias)
            .expressionAttributeValues(attrValues)
            .build();

        try {
            QueryResponse response = ddb.query(queryReq);
            return response.count();
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return -1;
    }
}

```

Mengkueri tabel menggunakan DynamoDbClient dan indeks sekunder.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */

```



```
* Create the Movies table by running the Scenario example and loading the Movie
* data from the JSON file. Next create a secondary
* index for the Movies table that uses only the year column. Name the index
* **year-index**. For more information, see:
*
* https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html
*/
public class QueryItemsUsingIndex {
    public static void main(String[] args) {
        String tableName = "Movies";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        queryIndex(ddb, tableName);
        ddb.close();
    }

    public static void queryIndex(DynamoDbClient ddb, String tableName) {
        try {
            Map<String, String> expressionAttributesNames = new HashMap<>();
            expressionAttributesNames.put("#year", "year");
            Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
            expressionAttributeValues.put(":yearValue",
                AttributeValue.builder().n("2013").build());

            QueryRequest request = QueryRequest.builder()
                .tableName(tableName)
                .indexName("year-index")
                .keyConditionExpression("#year = :yearValue")
                .expressionAttributeNames(expressionAttributesNames)
                .expressionAttributeValues(expressionAttributeValues)
                .build();

            System.out.println("=== Movie Titles ===");
            QueryResponse response = ddb.query(request);
            response.items()
                .forEach(movie -> System.out.println(movie.get("title").s()));

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

```
}  
}
```

- Untuk detail API, lihat [Query](#) di Referensi API AWS SDK for Java 2.x .

## Memindai tabel

Contoh kode berikut menunjukkan bagaimana untuk memindai tabel DynamoDB.

### SDK for Java 2.x

#### Note

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

Memindai tabel Amazon [DynamoDbClient](#)DynamoDB menggunakan.

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;  
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;  
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;  
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;  
import java.util.Map;  
import java.util.Set;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * To scan items from an Amazon DynamoDB table using the AWS SDK for Java V2,  
 * its better practice to use the  
 * Enhanced Client, See the EnhancedScanRecords example.  
 */
```

```
public class DynamoDBScanItems {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table to get information from
(for example, Music3).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        scanItems(ddb, tableName);
        ddb.close();
    }

    public static void scanItems(DynamoDbClient ddb, String tableName) {
        try {
            ScanRequest scanRequest = ScanRequest.builder()
                .tableName(tableName)
                .build();

            ScanResponse response = ddb.scan(scanRequest);
            for (Map<String, AttributeValue> item : response.items()) {
                Set<String> keys = item.keySet();
                for (String key : keys) {
                    System.out.println("The key name is " + key + "\n");
                    System.out.println("The value is " + item.get(key).s());
                }
            }
        }

        } catch (DynamoDbException e) {
```

```
        e.printStackTrace();
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [Scan](#) di Referensi API AWS SDK for Java 2.x .

## Memperbarui item dalam tabel

Contoh kode berikut menunjukkan cara memperbarui item dalam tabel DynamoDB.

### SDK for Java 2.x

#### Note

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

## Memperbarui item dalam tabel menggunakan [DynamoDbClient](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To update an Amazon DynamoDB table using the AWS SDK for Java V2, its better
 * practice to use the
```

```

* Enhanced Client, See the EnhancedModifyItem example.
*/
public class UpdateItem {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <key> <keyVal> <name> <updateVal>

            Where:
                tableName - The Amazon DynamoDB table (for example, Music3).
                key - The name of the key in the table (for example, Artist).
                keyVal - The value of the key (for example, Famous Band).
                name - The name of the column where the value is updated (for
example, Awards).
                updateVal - The value used to update an item (for example, 14).
            Example:
                UpdateItem Music3 Artist Famous Band Awards 14
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        String name = args[3];
        String updateVal = args[4];

        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
        updateTableItem(ddb, tableName, key, keyVal, name, updateVal);
        ddb.close();
    }

    public static void updateTableItem(DynamoDbClient ddb,
        String tableName,
        String key,
        String keyVal,
        String name,

```

```
String updateVal) {

    HashMap<String, AttributeValue> itemKey = new HashMap<>();
    itemKey.put(key, AttributeValue.builder()
        .s(keyVal)
        .build());

    HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
    updatedValues.put(name, AttributeValueUpdate.builder()
        .value(AttributeValue.builder().s(updateVal).build())
        .action(AttributeAction.PUT)
        .build());

    UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(itemKey)
        .attributeUpdates(updatedValues)
        .build();

    try {
        ddb.updateItem(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("The Amazon DynamoDB table was updated!");
}
```

- Untuk detail API, lihat [UpdateItem](#) di Referensi AWS SDK for Java 2.x API.

## Menulis sekumpulan item

Contoh kode berikut menunjukkan bagaimana menulis batch item DynamoDB.

## SDK for Java 2.x

### Note

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

## Menyisipkan banyak item ke dalam tabel dengan menggunakan klien layanan.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutRequest;
import software.amazon.awssdk.services.dynamodb.model.WriteRequest;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class BatchWriteItems {
    public static void main(String[] args){
        final String usage = ""

                Usage:
                <tableName>

                Where:
                tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
                .region(region)
                .build();

        addBatchItems(dynamoDbClient, tableName);
    }
}
```

```
public static void addBatchItems(DynamoDbClient dynamoDbClient, String
tableName) {
    // Specify the updates you want to perform.
    List<WriteRequest> writeRequests = new ArrayList<>();

    // Set item 1.
    Map<String, AttributeValue> item1Attributes = new HashMap<>();
    item1Attributes.put("Artist",
AttributeValue.builder().s("Artist1").build());
    item1Attributes.put("Rating", AttributeValue.builder().s("5").build());
    item1Attributes.put("Comments", AttributeValue.builder().s("Great
song!").build());
    item1Attributes.put("SongTitle",
AttributeValue.builder().s("SongTitle1").build());

writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item1Attribut

    // Set item 2.
    Map<String, AttributeValue> item2Attributes = new HashMap<>();
    item2Attributes.put("Artist",
AttributeValue.builder().s("Artist2").build());
    item2Attributes.put("Rating", AttributeValue.builder().s("4").build());
    item2Attributes.put("Comments", AttributeValue.builder().s("Nice
melody.").build());
    item2Attributes.put("SongTitle",
AttributeValue.builder().s("SongTitle2").build());

writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item2Attribut

    try {
        // Create the BatchWriteItemRequest.
        BatchWriteItemRequest batchWriteItemRequest =
BatchWriteItemRequest.builder()
            .requestItems(Map.of(tableName, writeRequests))
            .build();

        // Execute the BatchWriteItem operation.
        BatchWriteItemResponse batchWriteItemResponse =
dynamoDbClient.batchWriteItem(batchWriteItemRequest);

        // Process the response.
        System.out.println("Batch write successful: " + batchWriteItemResponse);

    } catch (DynamoDbException e) {
```



```
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

Menyisipkan banyak item ke dalam tabel menggunakan klien yang disempurnakan.

```
import com.example.dynamodb.Customer;
import com.example.dynamodb.Music;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.Key;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.enhanced.dynamodb.model.BatchWriteItemEnhancedRequest;
import software.amazon.awssdk.enhanced.dynamodb.model.WriteBatch;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/*
 * Before running this code example, create an Amazon DynamoDB table named Customer
 * with these columns:
 * - id - the id of the record that is the key
 * - custName - the customer name
 * - email - the email value
 * - registrationDate - an instant value when the item was added to the table
 *
 * Also, ensure that you have set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnhancedBatchWriteItems {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
```

```
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();
DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
    .dynamoDbClient(ddb)
    .build();
putBatchRecords(enhancedClient);
ddb.close();
}

public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient) {
    try {
        DynamoDbTable<Customer> customerMappedTable =
enhancedClient.table("Customer",
                        TableSchema.fromBean(Customer.class));
        DynamoDbTable<Music> musicMappedTable =
enhancedClient.table("Music",
                        TableSchema.fromBean(Music.class));
        LocalDate localDate = LocalDate.parse("2020-04-07");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        Customer record2 = new Customer();
        record2.setCustName("Fred Pink");
        record2.setId("id110");
        record2.setEmail("fredp@noserver.com");
        record2.setRegistrationDate(instant);

        Customer record3 = new Customer();
        record3.setCustName("Susan Pink");
        record3.setId("id120");
        record3.setEmail("spink@noserver.com");
        record3.setRegistrationDate(instant);

        Customer record4 = new Customer();
        record4.setCustName("Jerry orange");
        record4.setId("id101");
        record4.setEmail("jorange@noserver.com");
        record4.setRegistrationDate(instant);

        BatchWriteItemEnhancedRequest batchWriteItemEnhancedRequest
= BatchWriteItemEnhancedRequest
                                .builder()
```

```

        .writeBatches(

WriteBatch.builder(Customer.class) // add items to the Customer

    // table

    .mappedTableResource(customerMappedTable)

    .addPutItem(builder -> builder.item(record2))

    .addPutItem(builder -> builder.item(record3))

    .addPutItem(builder -> builder.item(record4))

                                                                    .build(),

WriteBatch.builder(Music.class) // delete an item from the Music

    // table

    .mappedTableResource(musicMappedTable)

    .addDeleteItem(builder -> builder.key(

        Key.builder().partitionValue(

            "Famous Band")

            .build()))

                                                                    .build())

                                                                    .build();

                                                                    // Add three items to the Customer table and delete one item
from the Music

                                                                    // table.

enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);
    System.out.println("done");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [BatchWriteItem](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

Memulai tabel, item, dan kueri

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- 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 for Java 2.x

### Note

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

Buat tabel DynamoDB.

```
// Create a table with a Sort key.
public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
```

```
        .attributeName("title")
        .attributeType("S")
        .build());

ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
KeySchemaElement key = KeySchemaElement.builder()
    .attributeName("year")
    .keyType(KeyType.HASH)
    .build();

KeySchemaElement key2 = KeySchemaElement.builder()
    .attributeName("title")
    .keyType(KeyType.RANGE)
    .build();

// Add KeySchemaElement objects to the list.
tableKey.add(key);
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(10L)
        .writeCapacityUnits(10L)
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " + newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
}
```

```

        System.exit(1);
    }
}

```

Buat fungsi pembantu untuk mengunduh dan mengekstrak file JSON sampel.

```

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

    DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {
        // Only add 200 Movies to the table.
        if (t == 200)
            break;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        Movies movies = new Movies();
        movies.setYear(year);
        movies.setTitle(title);
        movies.setInfo(info);

        // Put the data into the Amazon DynamoDB Movie table.
        mappedTable.putItem(movies);
        t++;
    }
}

```

## Dapatkan item dari tabel.

```
public static void getItem(DynamoDbClient ddb) {

    HashMap<String, AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put("year", AttributeValue.builder()
        .n("1933")
        .build());

    keyToGet.put("title", AttributeValue.builder()
        .s("King Kong")
        .build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName("Movies")
        .build();

    try {
        Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();

        if (returnedItem != null) {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");

            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
            }
        } else {
            System.out.format("No item found with the key %s!\n", "year");
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

## Contoh lengkap.

```
/**
```

```

* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This Java example performs these tasks:
*
* 1. Creates the Amazon DynamoDB Movie table with partition and sort key.
* 2. Puts data into the Amazon DynamoDB table from a JSON document using the
* Enhanced client.
* 3. Gets data from the Movie table.
* 4. Adds a new item.
* 5. Updates an item.
* 6. Uses a Scan to query items using the Enhanced client.
* 7. Queries all items where the year is 2013 using the Enhanced Client.
* 8. Deletes the table.
*/

```

```

public class Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <fileName>

            Where:
                fileName - The path to the moviedata.json file that you can
download from the Amazon DynamoDB Developer Guide.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = "Movies";
        String fileName = args[0];
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)

```



```
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon DynamoDB example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println(
        "1. Creating an Amazon DynamoDB table named Movies with a key named
year and a sort key named title.");
    createTable(ddb, tableName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Loading data into the Amazon DynamoDB table.");
    loadData(ddb, tableName, fileName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Getting data from the Movie table.");
    getItem(ddb);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Putting a record into the Amazon DynamoDB table.");
    putRecord(ddb);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("5. Updating a record.");
    updateTableItem(ddb, tableName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("6. Scanning the Amazon DynamoDB table.");
    scanMovies(ddb, tableName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("7. Querying the Movies released in 2013.");
    queryTable(ddb);
    System.out.println(DASHES);

    System.out.println(DASHES);
```

```
        System.out.println("8. Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        System.out.println(DASHES);

        ddb.close();
    }

    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE)
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .provisionedThroughput(ProvisionedThroughput.builder()
                .readCapacityUnits(10L)
                .writeCapacityUnits(10L)
                .build())
```

```
        .attributeDefinitions(attributeDefinitions)
        .tableName(tableName)
        .build();

    try {
        CreateTableResponse response = ddb.createTable(request);
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        // Wait until the Amazon DynamoDB table is created.
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        String newTable = response.tableDescription().tableName();
        System.out.println("The " + newTable + " was successfully created.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Query the table.
public static void queryTable(DynamoDbClient ddb) {
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        QueryConditional queryConditional = QueryConditional
            .keyEqualTo(Key.builder()
                .partitionValue(2013)
                .build());

        // Get items in the table and write out the ID value.
        Iterator<Movies> results =
custTable.query(queryConditional).items().iterator();
        String result = "";

        while (results.hasNext()) {
            Movies rec = results.next();
```

```
        System.out.println("The title of the movie is " + rec.getTitle());
        System.out.println("The movie information is " + rec.getInfo());
    }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Scan the table.
public static void scanMovies(DynamoDbClient ddb, String tableName) {
    System.out.println("***** Scanning all movies.\n");
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        Iterator<Movies> results = custTable.scan().items().iterator();
        while (results.hasNext()) {
            Movies rec = results.next();
            System.out.println("The movie title is " + rec.getTitle());
            System.out.println("The movie year is " + rec.getYear());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

    DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
```

```

    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {
        // Only add 200 Movies to the table.
        if (t == 200)
            break;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        Movies movies = new Movies();
        movies.setYear(year);
        movies.setTitle(title);
        movies.setInfo(info);

        // Put the data into the Amazon DynamoDB Movie table.
        mappedTable.putItem(movies);
        t++;
    }
}

// Update the record to include show only directors.
public static void updateTableItem(DynamoDbClient ddb, String tableName) {
    HashMap<String, AttributeValue> itemKey = new HashMap<>();
    itemKey.put("year", AttributeValue.builder().n("1933").build());
    itemKey.put("title", AttributeValue.builder().s("King Kong").build());

    HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
    updatedValues.put("info", AttributeValueUpdate.builder()
        .value(AttributeValue.builder().s("{\"directors\": [\"Merian C.
Cooper\", \"Ernest B. Schoedsack\"]")
            .build())
        .action(AttributeAction.PUT)
        .build());

    UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(itemKey)
        .attributeUpdates(updatedValues)

```

```
        .build();

    try {
        ddb.updateItem(request);
    } catch (ResourceNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    System.out.println("Item was updated!");
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    System.out.println(tableName + " was successfully deleted!");
}

public static void putRecord(DynamoDbClient ddb) {
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> table = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));

        // Populate the Table.
        Movies record = new Movies();
        record.setYear(2020);
        record.setTitle("My Movie2");
        record.setInfo("no info");
```

```
        table.putItem(record);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Added a new movie to the table.");
}

public static void getItem(DynamoDbClient ddb) {

    HashMap<String, AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put("year", AttributeValue.builder()
        .n("1933")
        .build());

    keyToGet.put("title", AttributeValue.builder()
        .s("King Kong")
        .build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName("Movies")
        .build();

    try {
        Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();

        if (returnedItem != null) {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");

            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
            }
        } else {
            System.out.format("No item found with the key %s!\n", "year");
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}  
}
```


- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

Melakukan kueri pada tabel menggunakan batch pernyataan PartiQL

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- 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 for Java 2.x

 Note

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

```
public class ScenarioPartiQLBatch {  
    public static void main(String[] args) throws IOException {
```



```
String tableName = "MoviesPartiQBatch";
Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();

System.out.println("***** Creating an Amazon DynamoDB table named
" + tableName
    + " with a key named year and a sort key named
title.");
createTable(ddb, tableName);

System.out.println("***** Adding multiple records into the " +
tableName
    + " table using a batch command.");
putRecordBatch(ddb);

System.out.println("***** Updating multiple records using a batch
command.");
updateTableItemBatch(ddb);

System.out.println("***** Deleting multiple records using a batch
command.");
deleteItemBatch(ddb);

System.out.println("***** Deleting the Amazon DynamoDB table.");
deleteDynamoDBTable(ddb, tableName);
ddb.close();
}

public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new
ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("title")
        .attributeType("S")
```

```
        .build());

    ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
    KeySchemaElement key = KeySchemaElement.builder()
        .attributeName("year")
        .keyType(KeyType.HASH)
        .build();

    KeySchemaElement key2 = KeySchemaElement.builder()
        .attributeName("title")
        .keyType(KeyType.RANGE) // Sort
        .build();

    // Add KeySchemaElement objects to the list.
    tableKey.add(key);
    tableKey.add(key2);

    CreateTableRequest request = CreateTableRequest.builder()
        .keySchema(tableKey)

        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10))
            .build())
        .attributeDefinitions(attributeDefinitions)
        .tableName(tableName)
        .build();

    try {
        CreateTableResponse response = ddb.createTable(request);
        DescribeTableRequest tableRequest =
DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        // Wait until the Amazon DynamoDB table is created.
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter
            .waitUntilTableExists(tableRequest);

        waiterResponse.matched().response().ifPresent(System.out::println);
        String newTable = response.tableDescription().tableName();
        System.out.println("The " + newTable + " was successfully
created.");
    }
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecordBatch(DynamoDbClient ddb) {
        String sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE
{'year':?, 'title' : ?, 'info' : ?}";
        try {
            // Create three movies to add to the Amazon DynamoDB table.
            // Set data for Movie 1.
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2022"))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("My Movie 1")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            BatchStatementRequest statementRequestMovie1 =
BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parameters)
                .build();

            // Set data for Movie 2.
            List<AttributeValue> parametersMovie2 = new ArrayList<>();
            AttributeValue attMovie2 = AttributeValue.builder()
                .n(String.valueOf("2022"))
                .build();

            AttributeValue attMovie2A = AttributeValue.builder()
```

```
        .s("My Movie 2")
        .build();

    AttributeValue attMovie2B = AttributeValue.builder()
        .s("No Information")
        .build();

    parametersMovie2.add(attMovie2);
    parametersMovie2.add(attMovie2A);
    parametersMovie2.add(attMovie2B);

    BatchStatementRequest statementRequestMovie2 =
BatchStatementRequest.builder()
        .statement(sqlStatement)
        .parameters(parametersMovie2)
        .build();

    // Set data for Movie 3.
    List<AttributeValue> parametersMovie3 = new ArrayList<>();
    AttributeValue attMovie3 = AttributeValue.builder()
        .n(String.valueOf("2022"))
        .build();

    AttributeValue attMovie3A = AttributeValue.builder()
        .s("My Movie 3")
        .build();

    AttributeValue attMovie3B = AttributeValue.builder()
        .s("No Information")
        .build();

    parametersMovie3.add(attMovie3);
    parametersMovie3.add(attMovie3A);
    parametersMovie3.add(attMovie3B);

    BatchStatementRequest statementRequestMovie3 =
BatchStatementRequest.builder()
        .statement(sqlStatement)
        .parameters(parametersMovie3)
        .build();

    // Add all three movies to the list.
    List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();
```

```
        myBatchStatementList.add(statementRequestMovie1);
        myBatchStatementList.add(statementRequestMovie2);
        myBatchStatementList.add(statementRequestMovie3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
                                .statements(myBatchStatementList)
                                .build();

        BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
        System.out.println("ExecuteStatement successful: " +
response.toString());
        System.out.println("Added new movies using a batch
command.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateTableItemBatch(DynamoDbClient ddb) {
    String sqlStatement = "UPDATE MoviesPartiQBatch SET info =
'directors\":[\"Merian C. Cooper\", \"Ernest B. Schoedsack' where year=? and
title=?";

    List<AttributeValue> parametersRec1 = new ArrayList<>();

    // Update three records.
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2022"))
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("My Movie 1")
        .build();

    parametersRec1.add(att1);
    parametersRec1.add(att2);

    BatchStatementRequest statementRequestRec1 =
BatchStatementRequest.builder()
                        .statement(sqlStatement)
                        .parameters(parametersRec1)
```

```
        .build();

// Update record 2.
List<AttributeValue> parametersRec2 = new ArrayList<>();
AttributeValue attRec2 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec2a = AttributeValue.builder()
    .s("My Movie 2")
    .build();

parametersRec2.add(attRec2);
parametersRec2.add(attRec2a);
BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec2)
    .build();

// Update record 3.
List<AttributeValue> parametersRec3 = new ArrayList<>();
AttributeValue attRec3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec3a = AttributeValue.builder()
    .s("My Movie 3")
    .build();

parametersRec3.add(attRec3);
parametersRec3.add(attRec3a);
BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec3)
    .build();

// Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();
myBatchStatementList.add(statementRequestRec1);
myBatchStatementList.add(statementRequestRec2);
myBatchStatementList.add(statementRequestRec3);
```

```
        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
                                .statements(myBatchStatementList)
                                .build();

        try {
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
            System.out.println("ExecuteStatement successful: " +
response.toString());
            System.out.println("Updated three movies using a batch
command.");
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }

    public static void deleteItemBatch(DynamoDbClient ddb) {
        String sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ?
and title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Specify three records to delete.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();
```

```
// Specify record 2.
List<AttributeValue> parametersRec2 = new ArrayList<>();
AttributeValue attRec2 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec2a = AttributeValue.builder()
    .s("My Movie 2")
    .build();

parametersRec2.add(attRec2);
parametersRec2.add(attRec2a);
BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec2)
    .build();

// Specify record 3.
List<AttributeValue> parametersRec3 = new ArrayList<>();
AttributeValue attRec3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec3a = AttributeValue.builder()
    .s("My Movie 3")
    .build();

parametersRec3.add(attRec3);
parametersRec3.add(attRec3a);

BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec3)
    .build();

// Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();
myBatchStatementList.add(statementRequestRec1);
myBatchStatementList.add(statementRequestRec2);
myBatchStatementList.add(statementRequestRec3);
```



```
        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();

        try {
            ddb.batchExecuteStatement(batchRequest);
            System.out.println("Deleted three movies using a batch
command.");
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName)
    {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    private static ExecuteStatementResponse
executeStatementRequest(DynamoDbClient ddb, String statement,
        List<AttributeValue> parameters) {
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
            .statement(statement)
            .parameters(parameters)
            .build();

        return ddb.executeStatement(request);
    }
}
```


- Untuk detail API, lihat [BatchExecuteStatement](#) di Referensi AWS SDK for Java 2.x API.

Melakukan kueri tabel menggunakan PartiQL

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- 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 for Java 2.x

 Note

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

```
public class ScenarioPartiQ {
    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <fileName>

            Where:
                fileName - The path to the moviedata.json file that you can
download from the Amazon DynamoDB Developer Guide.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String fileName = args[0];
        String tableName = "MoviesPartiQ";
```

```
Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();

System.out.println(
    "***** Creating an Amazon DynamoDB table named MoviesPartiQ with a
key named year and a sort key named title.");
createTable(ddb, tableName);

System.out.println("***** Loading data into the MoviesPartiQ table.");
loadData(ddb, fileName);

System.out.println("***** Getting data from the MoviesPartiQ table.");
getItem(ddb);

System.out.println("***** Putting a record into the MoviesPartiQ table.");
putRecord(ddb);

System.out.println("***** Updating a record.");
updateTableItem(ddb);

System.out.println("***** Querying the movies released in 2013.");
queryTable(ddb);

System.out.println("***** Deleting the Amazon DynamoDB table.");
deleteDynamoDBTable(ddb, tableName);
ddb.close();
}

public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("title")
        .attributeType("S")
        .build());
```

```
ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
KeySchemaElement key = KeySchemaElement.builder()
    .attributeName("year")
    .keyType(KeyType.HASH)
    .build();

KeySchemaElement key2 = KeySchemaElement.builder()
    .attributeName("title")
    .keyType(KeyType.RANGE) // Sort
    .build();

// Add KeySchemaElement objects to the list.
tableKey.add(key);
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(new Long(10))
        .writeCapacityUnits(new Long(10))
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " + newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

```
// Load data into the table.
public static void loadData(DynamoDbClient ddb, String fileName) throws
IOException {

    String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    List<AttributeValue> parameters = new ArrayList<>();
    while (iter.hasNext()) {

        // Add 200 movies to the table.
        if (t == 200)
            break;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf(year))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s(title)
            .build();

        AttributeValue att3 = AttributeValue.builder()
            .s(info)
            .build();

        parameters.add(att1);
        parameters.add(att2);
        parameters.add(att3);

        // Insert the movie into the Amazon DynamoDB table.
        executeStatementRequest(ddb, sqlStatement, parameters);
        System.out.println("Added Movie " + title);
    }
}
```

```
        parameters.remove(att1);
        parameters.remove(att2);
        parameters.remove(att3);
        t++;
    }
}

public static void getItem(DynamoDbClient ddb) {

    String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n("2012")
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("The Perks of Being a Wallflower")
        .build();

    parameters.add(att1);
    parameters.add(att2);

    try {
        ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
        System.out.println("ExecuteStatement successful: " +
response.toString());

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void putRecord(DynamoDbClient ddb) {

    String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
    try {
        List<AttributeValue> parameters = new ArrayList<>();

        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2020"))
```

```
        .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie")
            .build();

        AttributeValue att3 = AttributeValue.builder()
            .s("No Information")
            .build();

        parameters.add(att1);
        parameters.add(att2);
        parameters.add(att3);

        executeStatementRequest(ddb, sqlStatement, parameters);
        System.out.println("Added new movie.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateTableItem(DynamoDbClient ddb) {

    String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian C. Cooper\", \"Ernest B. Schoedsack' where year=? and title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2013"))
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("The East")
        .build();

    parameters.add(att1);
    parameters.add(att2);

    try {
        executeStatementRequest(ddb, sqlStatement, parameters);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
    }
}
```

```
        System.exit(1);
    }
    System.out.println("Item was updated!");
}

// Query the table where the year is 2013.
public static void queryTable(DynamoDbClient ddb) {
    String sqlStatement = "SELECT * FROM MoviesPartiQ where year = ? ORDER BY
year";
    try {

        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2013"))
            .build();
        parameters.add(att1);

        // Get items in the table and write out the ID value.
        ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
        System.out.println("ExecuteStatement successful: " +
response.toString());

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}
```



```
private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient
ddb, String statement,
    List<AttributeValue> parameters) {
    ExecuteStatementRequest request = ExecuteStatementRequest.builder()
        .statement(statement)
        .parameters(parameters)
        .build();

    return ddb.executeStatement(request);
}

private static void processResults(ExecuteStatementResponse
executeStatementResult) {
    System.out.println("ExecuteStatement successful: " +
executeStatementResult.toString());
}
}
```

- Untuk detail API, lihat [ExecuteStatement](#) di Referensi AWS SDK for Java 2.x API.

## Contoh Amazon EC2 menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan Amazon EC2. AWS SDK for Java 2.x

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

### Ayo Memulai

#### Halo Amazon EC2

Contoh kode berikut ini menunjukkan cara mendapatkan data tentang tipe instans Amazon EC2.

## SDK for Java 2.x

 Note

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

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
            .forEach(group -> System.out
                .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DescribeSecurityGroups](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

### Mengalokasikan alamat IP Elastis

Contoh kode berikut menunjukkan cara mengalokasikan alamat IP Elastis untuk Amazon EC2.

SDK for Java 2.x

#### Note

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

```
public static String allocateAddress(Ec2Client ec2) {
    try {
        AllocateAddressRequest allocateRequest =
AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        AllocateAddressResponse allocateResponse =
ec2.allocateAddress(allocateRequest);
        return allocateResponse.allocationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Untuk detail API, lihat [AllocateAddress](#) di Referensi AWS SDK for Java 2.x API.

### Mengaitkan alamat IP Elastis dengan instans

Contoh kode berikut menunjukkan cara mengaitkan alamat IP Elastis dengan instans Amazon EC2.

## SDK for Java 2.x

 Note

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

```
public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
    try {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
        return associateResponse.associationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Untuk detail API, lihat [AssociateAddress](#) di Referensi AWS SDK for Java 2.x API.

## Membuat grup keamanan

Contoh kode berikut menunjukkan cara membuat grup keamanan Amazon EC2.

## SDK for Java 2.x

 Note

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

```
public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupName)
```

```

        .ipPermissions(ipPerm, ipPerm2)
        .build();

        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.println("Successfully added ingress policy to security group
" + groupName);
        return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

```

- Untuk detail API, lihat [CreateSecurityGroup](#) di Referensi AWS SDK for Java 2.x API.

## Membuat pasangan kunci

Contoh kode berikut menunjukkan cara membuat security key pair untuk Amazon EC2.

SDK for Java 2.x

### Note

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

```

public static void createKeyPair(Ec2Client ec2, String keyName, String fileName)
{
    try {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();

        CreateKeyPairResponse response = ec2.createKeyPair(request);
        String content = response.keyMaterial();
        BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
        writer.write(content);
        writer.close();
    }
}

```

```
        System.out.println("Successfully created key pair named " + keyName);
    } catch (Ec2Exception | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [CreateKeyPair](#) di Referensi AWS SDK for Java 2.x API.

## Membuat dan menjalankan instans

Contoh kode berikut menunjukkan cara membuat dan menjalankan instans Amazon EC2.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This code example requires an AMI value. You can learn more about this value
 * by reading this documentation topic:
```

```
*
* https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/AMIs.html
*/
public class CreateInstance {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <name> <amiId>

            Where:
                name - An instance name value that you can obtain from the AWS
Console (for example, ami-xxxxxx5c8b987b1a0).\s
                amiId - An Amazon Machine Image (AMI) value that you can obtain
from the AWS Console (for example, i-xxxxxx2734106d0ab).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String amiId = args[1];
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        String instanceId = createEC2Instance(ec2, name, amiId);
        System.out.println("The Amazon EC2 Instance ID is " + instanceId);
        ec2.close();
    }

    public static String createEC2Instance(Ec2Client ec2, String name, String amiId)
    {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .imageId(amiId)
            .instanceType(InstanceType.T1_MICRO)
            .maxCount(1)
            .minCount(1)
            .build();

        // Use a waiter to wait until the instance is running.
    }
}
```



```
System.out.println("Going to start an EC2 instance using a waiter");
RunInstancesResponse response = ec2.runInstances(runRequest);
String instanceIdVal = response.instances().get(0).instanceId();
ec2.waiter().waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal));
Tag tag = Tag.builder()
    .key("Name")
    .value(name)
    .build();

CreateTagsRequest tagRequest = CreateTagsRequest.builder()
    .resources(instanceIdVal)
    .tags(tag)
    .build();

try {
    ec2.createTags(tagRequest);
    System.out.printf("Successfully started EC2 Instance %s based on AMI
%s", instanceIdVal, amiId);
    return instanceIdVal;

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return "";
}
```

- Untuk detail API, lihat [RunInstances](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus grup keamanan

Contoh kode berikut menunjukkan cara menghapus grup keamanan Amazon EC2.

### SDK for Java 2.x

#### Note

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

```
public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
    try {
        DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
        .groupId(groupId)
        .build();

        ec2.deleteSecurityGroup(request);
        System.out.println("Successfully deleted security group with Id " +
groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DeleteSecurityGroup](#) di Referensi AWS SDK for Java 2.x API.

## Membuat pasangan kunci keamanan

Contoh kode berikut menunjukkan cara menghapus key pair keamanan Amazon EC2.

### SDK for Java 2.x

#### Note

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

```
public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
        .keyName(keyPair)
        .build();

        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named " + keyPair);
    }
}
```

```
    } catch (Ec2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Untuk detail API, lihat [DeleteKeyPair](#) di Referensi AWS SDK for Java 2.x API.

## Menjelaskan instans

Contoh kode berikut menunjukkan cara mendeskripsikan instans Amazon EC2.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.ec2.Ec2Client;  
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;  
import software.amazon.awssdk.services.ec2.model.Ec2Exception;  
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesIterable;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class DescribeInstances {  
    public static void main(String[] args) {  
        Region region = Region.US_EAST_1;  
        Ec2Client ec2 = Ec2Client.builder()  
            .region(region)  
            .build();
```

```
        describeEC2Instances(ec2);
        ec2.close();
    }

    public static void describeEC2Instances(Ec2Client ec2) {
        try {
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                .maxResults(10)
                .build();

            DescribeInstancesIterable instancesIterable =
ec2.describeInstancesPaginator(request);
            instancesIterable.stream()
                .flatMap(r -> r.reservations().stream())
                .flatMap(reservation -> reservation.instances().stream())
                .forEach(instance -> {
                    System.out.println("Instance Id is " + instance.instanceId());
                    System.out.println("Image id is " + instance.imageId());
                    System.out.println("Instance type is " +
instance.instanceType());
                    System.out.println("Instance state name is " +
instance.state().name());
                    System.out.println("Monitoring information is " +
instance.monitoring().state());
                });


        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorCode());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DescribeInstances](#) di Referensi AWS SDK for Java 2.x API.

## Memisahkan alamat IP Elastis dari instans

Contoh kode berikut menunjukkan cara memisahkan alamat IP Elastis dari instans Amazon EC2.

## SDK for Java 2.x

 Note

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

```
public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
        DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();


        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DisassociateAddress](#) di Referensi AWS SDK for Java 2.x API.

## Mendapatkan data tentang grup keamanan

Contoh kode berikut menunjukkan cara mendapatkan data tentang grup keamanan Amazon EC2.

## SDK for Java 2.x

 Note

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

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
```

```

    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
        .groupIds(groupId)
        .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
            .forEach(group -> System.out
                .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Untuk detail API, lihat [DescribeSecurityGroups](#) di Referensi AWS SDK for Java 2.x API.

## Mendapatkan data tentang tipe instans

Contoh kode berikut menunjukkan cara mendapatkan data tentang jenis instans Amazon EC2.

SDK for Java 2.x

### Note

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

```

// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType;
    try {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()

```

```
        .maxResults(10)
        .build();

    DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
    List<InstanceTypeInfo> instanceTypes = response.getInstanceTypes();
    for (InstanceTypeInfo type : instanceTypes) {
        System.out.println("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
        System.out.println("Network information is " +
type.networkInfo().toString());
        System.out.println("Instance type is " +
type.getInstanceType().toString());
        instanceType = type.getInstanceType().toString();
        if (instanceType.compareTo("t2.2xlarge") == 0){
            return instanceType;
        }
    }

} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
```

- Untuk detail API, lihat [DescribeInstanceTypes](#) di Referensi AWS SDK for Java 2.x API.

## Membuat daftar pasangan kunci keamanan

Contoh kode berikut menunjukkan cara mencantumkan pasangan kunci keamanan Amazon EC2.

### SDK for Java 2.x

#### Note

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

```
public static void describeKeys(Ec2Client ec2) {
```

```
try {
    DescribeKeyPairsResponse response = ec2.describeKeyPairs();
    response.keyPairs().forEach(keyPair -> System.out.printf(
        "Found key pair with name %s " +
        "and fingerprint %s",
        keyPair.keyName(),
        keyPair.keyFingerprint()));
} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

- Untuk detail API, lihat [DescribeKeyPairs](#) di Referensi AWS SDK for Java 2.x API.

## Merilis alamat IP Elastis

Contoh kode berikut menunjukkan cara melepaskan alamat IP Elastis.

### SDK for Java 2.x

#### Note

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

```
public static void releaseEC2Address(Ec2Client ec2, String allocId) {
    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        ec2.releaseAddress(request);
        System.out.println("Successfully released Elastic IP address " +
            allocId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



```
}
```

- Untuk detail API, lihat [ReleaseAddress](#) di Referensi AWS SDK for Java 2.x API.

## Menetapkan aturan masuk untuk grup keamanan

Contoh kode berikut menunjukkan cara menetapkan aturan masuk untuk grup keamanan Amazon EC2.

### SDK for Java 2.x

#### Note

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

```
public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();
```

```

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupName)
            .ipPermissions(ipPerm, ipPerm2)
            .build();

        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.println("Successfully added ingress policy to security group
" + groupName);
        return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

```

- Untuk detail API, lihat [AuthorizeSecurityGroupIngress](#) di Referensi AWS SDK for Java 2.x API.

## Memulai instans

Contoh kode berikut menunjukkan cara memulai instans Amazon EC2.

## SDK for Java 2.x

### Note

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

```
public static void startInstance(Ec2Client ec2, String instanceId) {
```

```

Ec2Waiter ec2Waiter = Ec2Waiter.builder()
    .overrideConfiguration(b -> b.maxAttempts(100))
    .client(ec2)
    .build();

StartInstancesRequest request = StartInstancesRequest.builder()
    .instanceIds(instanceId)
    .build();

System.out.println("Use an Ec2Waiter to wait for the instance to run. This
will take a few minutes.");
ec2.startInstances(request);
DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
    .instanceIds(instanceId)
    .build();

WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Successfully started instance " + instanceId);
}

```

- Untuk detail API, lihat [StartInstances](#) di Referensi AWS SDK for Java 2.x API.

## Menghentikan instans

Contoh kode berikut menunjukkan cara menghentikan instans Amazon EC2.

SDK for Java 2.x

### Note

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

```

public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)

```

```
        .build();
    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully stopped instance " + instanceId);
}
```

- Untuk detail API, lihat [StopInstances](#) di Referensi AWS SDK for Java 2.x API.

## Akhiri instans

Contoh kode berikut menunjukkan cara menghentikan instans Amazon EC2.

SDK for Java 2.x

### Note

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

```
public static void terminateEC2(Ec2Client ec2, String instanceId) {
    try {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
            .overrideConfiguration(b -> b.maxAttempts(100))
            .client(ec2)
            .build();
    }
}
```

```

        TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to
        terminate. This will take a few minutes.");
        ec2.terminateInstances(ti);
        DescribeInstancesRequest instanceRequest =
        DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
            .waitUntilInstanceTerminated(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
        System.out.println(instanceId + " is terminated!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Untuk detail API, lihat [TerminateInstances](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

### Membangun dan mengelola layanan yang tangguh

Contoh kode berikut menunjukkan cara membuat layanan web load-balanced yang mengembalikan rekomendasi buku, film, dan lagu. Contoh ini menunjukkan cara layanan tersebut merespons kegagalan, serta cara merestrukturisasi layanan agar lebih tangguh ketika terjadi kegagalan.

- Menggunakan grup Amazon EC2 Auto Scaling untuk membuat instans Amazon Elastic Compute Cloud (Amazon EC2) berdasarkan templat peluncuran dan menyimpan sejumlah instans dalam rentang yang ditentukan.
- Menangani dan mendistribusikan permintaan HTTP dengan Elastic Load Balancing.
- Memantau kondisi instans dalam grup Auto Scaling dan meneruskan permintaan hanya ke instans yang sehat.

- Menjalankan server web Python pada setiap instans EC2 untuk menangani permintaan HTTP. Server web merespons dengan memberikan rekomendasi dan melakukan pemeriksaan kondisi.
- Menyimulasikan layanan yang direkomendasikan dengan tabel Amazon DynamoDB.
- Kontrol respons server web terhadap permintaan dan pemeriksaan kesehatan dengan memperbarui AWS Systems Manager parameter.

## SDK for Java 2.x

### Note

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

Menjalankan skenario interaktif di prompt perintah.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\\
\\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\\
\\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
```

```
public static final String autoScalingGroupName = "doc-example-resilience-
group";
public static final String lbName = "doc-example-resilience-lb";
public static final String protocol = "HTTP";
public static final int port = 80;

public static final String DASHES = new String(new char[80]).replace("\0", "-");

public static void main(String[] args) throws IOException, InterruptedException
{
    Scanner in = new Scanner(System.in);
    Database database = new Database();
    AutoScaler autoScaler = new AutoScaler();
    LoadBalancer loadBalancer = new LoadBalancer();

    System.out.println(DASHES);
    System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("A - SETUP THE RESOURCES");
    System.out.println("Press Enter when you're ready to start deploying
resources.");
    in.nextLine();
    deploy(loadBalancer);
    System.out.println(DASHES);
    System.out.println(DASHES);
    System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    demo(loadBalancer);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("C - DELETE THE RESOURCES");
    System.out.println("""
        This concludes the demo of how to build and manage a resilient
service.

        To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
that were created for this demo.
        """);
```

```

System.out.println("\n Do you want to delete the resources (y/n)? ");
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

if (userInput.equals("y")) {
    // Delete resources here
    deleteResources(loadBalancer, autoScaler, database);
    System.out.println("Resources deleted.");
} else {
    System.out.println("""
        Okay, we'll leave the resources intact.
        Don't forget to delete them when you're done with them or you
might incur unexpected charges.
        """);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The example has completed. ");
System.out.println("\n Thanks for watching!");
System.out.println(DASHES);
}

// Deletes the AWS resources used in this example.
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
    throws IOException, InterruptedException {
    loadBalancer.deleteLoadBalancer(lbName);
    System.out.println("*** Wait 30 secs for resource to be deleted");
    TimeUnit.SECONDS.sleep(30);
    loadBalancer.deleteTargetGroup(targetGroupName);
    autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
    autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
    autoScaler.deleteTemplate(templateName);
    database.deleteTable(tableName);
}

private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
    System.out.println(
        """
                For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources

```



```

        to set up a load-balanced web service endpoint and explore
some ways to make it resilient
        against various kinds of failures.

```

```

        Some of the resources create by this demo are:

```

```

        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.

```

```

        \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.

```

```

        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.

```

```

        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.

```

```

        """);

```

```

        System.out.println("Press Enter when you're ready.");

```

```

        in.nextLine();

```

```

        System.out.println(DASHES);

```

```

        System.out.println(DASHES);

```

```

        System.out.println("Creating and populating a DynamoDB table named " +
tableName);

```

```

        Database database = new Database();

```

```

        database.createTable(tableName, fileName);

```

```

        System.out.println(DASHES);

```

```

        System.out.println(DASHES);

```

```

        System.out.println("""

```

```

                Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.

```

```

                This script starts a Python web server defined in the `server.py`
script. The web server

```

```

                listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.

```

```

                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged credentials.

```

```

                The template also defines an IAM policy that each instance uses to
assume a role that grants

```

```

                permissions to access the DynamoDB recommendation table and Systems
Manager parameters

```

```

                that control the flow of the demo.

```

```

        """);

```

```
LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(
    "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
System.out.println("*** Wait 30 secs for the VPC to be created");
TimeUnit.SECONDS.sleep(30);
AutoScaler autoScaler = new AutoScaler();
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

System.out.println("""
    At this point, you have EC2 instances created. Once each instance
starts, it listens for
    HTTP requests. You can see these instances in the console or
continue with the demo.
    Press Enter when you're ready to continue.
    """);

in.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Creating variables that control the flow of the demo.");
ParameterHelper paramHelper = new ParameterHelper();
paramHelper.reset();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an Elastic Load Balancing target group and load balancer.
The target group
    defines how the load balancer connects to instances. The load
balancer provides a
    single endpoint where clients connect and dispatches requests to
instances in the group.
    """);

String vpcId = autoScaler.getDefaultVPC();
```

```
List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
System.out.println("Verifying access to the load balancer endpoint...");
boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
if (!wasSuccessful) {
    System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to "http://checkip.amazonaws.com"
   HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
    try {
        // Execute the request and get the response
        HttpResponse response = httpClient.execute(httpGet);

        // Read the response content.
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

        // Print the public IP address.
        System.out.println("Public IP Address: " + ipAddress);
        GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
        if (!groupInfo.isPortOpen()) {
            System.out.println("""
                For this example to work, the default security group for
your default VPC must
                allow access from this computer. You can either add it
automatically from this
                example or add it yourself using the AWS Management
Console.
                """);

            System.out.println(
                "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
```

```

        System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
        String ans = in.nextLine();
        if ("y".equalsIgnoreCase(ans)) {
            autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
            System.out.println("Security group rule added.");
        } else {
            System.out.println("No security group rule added.");
        }
    }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
} else if (wasSuccessful) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" + elbDnsName);
} else {
    System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
    System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
    System.out.println("you can successfully make a GET request to the load
balancer.");
}

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();
    System.out.println("Read the ssm_only_policy.json file");
    String ssmOnlyPolicy = readFileAsString(ssmJSON);

    System.out.println("Resetting parameters to starting values for demo.");
    paramHelper.reset();

    System.out.println(

```

```
        ""
        This part of the demonstration shows how to toggle
different parts of the system
        to create situations where the web service fails, and shows
how using a resilient
        architecture can keep the web service running in spite of
these failures.

        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
        """);
demoChoices(loadBalancer);

System.out.println(
    ""
        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
        """);
paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

System.out.println(
    ""
        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
        """);
demoChoices(loadBalancer);

System.out.println(
    ""
        Instead of failing when the recommendation service fails,
the web service can return a static response.
        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
        """);
paramHelper.put(paramHelper.failureResponse, "static");

System.out.println("""
```

```
        Now, sending a GET request to the load balancer endpoint returns a
static response.
        The service still reports as healthy because health checks are still
shallow.
        """);
demoChoices(loadBalancer);

System.out.println("Let's reinstate the recommendation service.");
paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

System.out.println("""
        Let's also substitute bad credentials for one of the instances in
the target group so that it can't
        access the DynamoDB recommendation table. We will get an instance id
value.
        """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
AutoScaler autoScaler = new AutoScaler();

// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " + profileAssociationId);
System.out.println("Replacing the profile for instance " + badInstanceId
        + " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
        """)
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
        """);

demoChoices(loadBalancer);
```

```
System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
    the web service can access the DynamoDB table that it depends on for
recommendations. Note that
    the deep health check is only for ELB routing and not for Auto
Scaling instance health.
    This kind of deep health check is not recommended for Auto Scaling
instance health, because it
    risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
    """);

System.out.println("""
    By implementing deep health checks, the load balancer can detect
when one of the instances is failing
    and take that instance out of rotation.
    """);

paramHelper.put(paramHelper.healthCheck, "deep");

System.out.println("""
    Now, checking target health indicates that the instance with bad
credentials
    is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
    instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
    the load balancer takes unhealthy instances out of its rotation.
    """);

demoChoices(loadBalancer);

System.out.println(
    """
        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
        instance is to terminate it and let the auto scaler start a
new instance to replace it.
        """);
autoScaler.terminateInstance(badInstanceId);

System.out.println("""
```

Even while the instance is terminating and the new instance is starting, sending a GET request to the web service continues to get a successful recommendation response because the load balancer routes requests to the healthy instances. After the replacement instance starts and reports as healthy, it is included in the load balancing rotation.

Note that terminating and replacing an instance typically takes several minutes, during which time you can see the changing health check status until the new instance is running and healthy.

```

        """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
    InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();

```



```
        System.out.println("-".repeat(88));

        switch (choice) {
            case 0 -> {
                System.out.println("Request:\n");
                System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                CloseableHttpClient httpClient =
loadBalancer.createDefault();

                // Create an HTTP GET request to the ELB.
                HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);

                // Display the JSON response
                BufferedReader reader = new BufferedReader(
                    new
loadBalancer.getInputStream(response.getEntity().getContent()));
                StringBuilder jsonResponse = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null) {
                    jsonResponse.append(line);
                }
                reader.close();

                // Print the formatted JSON response.
                System.out.println("Full Response:\n");
                System.out.println(jsonResponse.toString());

                // Close the HTTP client.
                httpClient.close();
            }
            case 1 -> {
                System.out.println("\nChecking the health of load balancer
targets:\n");

                List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                for (TargetHealthDescription target : health) {
```

```

        System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
    }
    System.out.println("""
check to update
                                Note that it can take a minute or two for the health
                                after changes are made.
                                """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value between
0-2. Please select again.");
    }

    } catch (java.util.InputMismatchException e) {
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}

```

Membuat kelas yang menggabungkan tindakan Penskalaan Otomatis dan Amazon EC2.

```

public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;
}

```

```
private IamClient getIAMClient() {
    if (iamClient == null) {
        iamClient = IamClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return iamClient;
}

private SsmClient getSSMClient() {
    if (ssmClient == null) {
        ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ssmClient;
}

private Ec2Client getEc2Client() {
    if (ec2Client == null) {
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
    TerminateInstanceInAutoScalingGroupRequest
```

```
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
    software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
        .builder()
        .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                // name.
        .build();

    // Replace the IAM instance profile association for the EC2 instance.
    ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
        .builder()
        .iamInstanceProfile(iamInstanceProfile)
        .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
        .build();

    try {
        getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
        // Handle the response as needed.
    } catch (Ec2Exception e) {
        // Handle exceptions, log, or report the error.
    }
}
```

```

        System.err.println("Error: " + e.getMessage());
    }
    System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
        newInstanceProfileName);
    TimeUnit.SECONDS.sleep(15);
    boolean instReady = false;
    int tries = 0;

    // Reboot after 60 seconds
    while (!instReady) {
        if (tries % 6 == 0) {
            getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                .instanceIds(instanceId)
                .build());
            System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
        }
        tries++;
        try {
            TimeUnit.SECONDS.sleep(10);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
        List<InstanceInformation> instanceInformationList =
informationResponse.getInstanceInformationList();
        for (InstanceInformation info : instanceInformationList) {
            if (info.getInstanceId().equals(instanceId)) {
                instReady = true;
                break;
            }
        }
    }

    SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
            Collections.singletonList("cd / && sudo python3 server.py
80")))
        .build();

```

```
        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(secGroupId)
            .cidrIp(ipAddress)
            .fromPort(Integer.parseInt(port))
            .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
            .builder()
            .instanceProfileName(profileName)
            .build();

            GetInstanceProfileResponse response =
getInstanceProfileRequest;
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .roleName(roleName)
            .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        }
    }
}
```

```
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
        .roleName(roleName)
        .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
        .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(attachedPolicy.policyArn())
            .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");

    } catch (IamException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
```

```
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
    .autoScalingGroupName(groupName)
    .forceDelete(true)
    .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}

/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network, you
 * must instead specify a prefix list ID. You can also temporarily open the port
 * to
 * any IP address while running this example. If you do, be sure to remove
 * public
 * access when you're done.
 *
 */
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
    .filters(filter, filter1)
    .build();

        DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
            .describeSecurityGroups(securityGroupsRequest);
    }
}
```



```

        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " + secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " + ipPermission);
                    for (IpRange ipRange : ipPermission.ipRanges()) {
                        String cidrIp = ipRange.cidrIp();
                        if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                            System.out.println(cidrIp + " is applicable");
                            portIsOpen = true;
                        }
                    }

                    if (!ipPermission.prefixListIds().isEmpty()) {
                        System.out.println("Prefix lList is applicable");
                        portIsOpen = true;
                    }

                    if (!portIsOpen) {
                        System.out
                            .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                    } else {
                        break;
                    }
                }
            }
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }

    groupInfo.setPortOpen(portIsOpen);
    return groupInfo;
}

```

```
/*
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

    // Get availability zones.
    software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
        .builder()
        .build();

    DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
    List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

        .map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
```

```
        .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
        .launchTemplateName(templateName)
        .version("$Default")
        .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
        .launchTemplate(specification)
        .availabilityZones(zones)
        .maxSize(groupSize)
        .minSize(groupSize)
        .autoScalingGroupName(autoScalingGroupName)
        .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();
```

```
        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
            .name("vpc-id")
            .values(vpcId)
            .build();

        Filter azFilter = Filter.builder()
            .name("availability-zone")
            .values(availabilityZones)
            .build();

        Filter defaultForAZ = Filter.builder()
            .name("default-for-az")
            .values("true")
            .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
            .filters(vpcFilter, azFilter, defaultForAZ)
            .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
            .map(instance -> instance.instanceId())
            .collect(Collectors.toList());
    }
}
```

```
String[] instanceIdArray = instanceIds.toArray(new String[0]);
for (String instanceId : instanceIdArray) {
    System.out.println("Instance ID: " + instanceId);
    return instanceId;
}
return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
    ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
    ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
    for (Policy policy : listPoliciesResponse.policies()) {
        if (policy.policyName().equals(policyName)) {
            // List the entities (users, groups, roles) that are attached to the
policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
        .builder()
        .policyArn(policy.arn())
        .build();
```

```

        ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
            .listEntitiesForPolicy(listEntitiesRequest);
        if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
            || !listEntitiesResponse.policyRoles().isEmpty()) {
            // Detach the policy from any entities it is attached to.
            DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
                .policyArn(policy.arn())
                .roleName(roleName) // Specify the name of the IAM role
                .build();

            getIAMClient().detachRolePolicy(detachPolicyRequest);
            System.out.println("Policy detached from entities.");
        }

        // Now, you can delete the policy.
        DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
            .policyArn(policy.arn())
            .build();

        getIAMClient().deletePolicy(deletePolicyRequest);
        System.out.println("Policy deleted successfully.");
        break;
    }
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .roleName(roleName) // Remove the extra dot here
        .build();

```

```

        getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
        System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
    }

    // Delete the instance profile after removing all roles
    DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .build();

    getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
    System.out.println(InstanceProfile + " Deleted");
    System.out.println("All roles and policies are deleted.");
}
}

```

Membuat kelas yang menggabungkan tindakan Penyeimbangan Beban Elastis.

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
            .names(targetGroupName)
            .build();
    }
}

```

```
DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
    .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
    .build();

DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```



```
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }
        }
    }
}
```

```

    } catch (org.apache.http.conn.HttpHostConnectException e) {
        System.out.println(e.getMessage());
    }

    System.out.println("Status.." + success);
    return success;
}

/**
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}

/**
 * Creates an Elastic Load Balancing load balancer that uses the specified
 * subnets
 * and forwards requests to the specified target group.
 */
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,

```

```
String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()
```

```

        .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}

```

Membuat kelas yang menggunakan DynamoDB untuk mensimulasikan layanan yang direkomendasikan.

```

public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {

```

```

        // Describe the table and catch any exceptions.
        DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        getDynamoDbClient().describeTable(describeTableRequest);
        System.out.println("Table '" + tableName + "' exists.");
        return true;

    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " + e.getMessage());
    }
    return false;
}

/**
 * Creates a DynamoDB table to use a recommendation service. The table has a
 * hash key named 'MediaType' that defines the type of media recommended, such
 * as
 * Book or Movie, and a range key named 'ItemId' that, combined with the
 * MediaType,
 * forms a unique identifier for the recommended item.
 */
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()

```

```

        .attributeName("MediaType")
        .keyType(KeyType.HASH)
        .build(),
        KeySchemaElement.builder()
        .attributeName("ItemId")
        .keyType(KeyType.RANGE)
        .build()
    ).provisionedThroughput(
        ProvisionedThroughput.builder()
        .readCapacityUnits(5L)
        .writeCapacityUnits(5L)
        .build()
    ).build();

    getDynamoDbClient().createTable(createTableRequest);
    System.out.println("Creating table " + tableName + "...");

    // Wait until the Amazon DynamoDB table is created.
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    WaiterResponse<DescribeTableResponse> waiterResponse =
    dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Table " + tableName + " created.");

    // Add records to the table.
    populateTable(fileName, tableName);
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();

```

```

    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
    System.out.println("Added all records to the " + tableName);
}
}

```

Membuat kelas yang mengabungkan tindakan Systems Manager.

```

public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {

```

```
SsmClient ssmClient = SsmClient.builder()
    .region(Region.US_EAST_1)
    .build();

PutParameterRequest parameterRequest = PutParameterRequest.builder()
    .name(name)
    .value(value)
    .overwrite(true)
    .type("String")
    .build();

ssmClient.putParameter(parameterRequest);
System.out.printf("Setting demo parameter %s to '%s'.", name, value);
}
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
  - [AttachLoadBalancerTargetGroups](#)
  - [CreateAutoScalingGroup](#)
  - [CreateInstanceProfile](#)
  - [CreateLaunchTemplate](#)
  - [CreateListener](#)
  - [CreateLoadBalancer](#)
  - [CreateTargetGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DeleteInstanceProfile](#)
  - [DeleteLaunchTemplate](#)
  - [DeleteLoadBalancer](#)
  - [DeleteTargetGroup](#)
  - [DescribeAutoScalingGroups](#)
  - [DescribeAvailabilityZones](#)
  - [DescribeInstanceProfileAssociations](#)
  - [DescribeInstances](#)
  - [DescribeLoadBalancers](#)
  - [DescribeSubnets](#)



- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacelamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

## Memulai instans

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- 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 for Java 2.x

### Note

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

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs the following tasks:
 *
```

```

* 1. Creates an RSA key pair and saves the private key data as a .pem file.
* 2. Lists key pairs.
* 3. Creates a security group for the default VPC.
* 4. Displays security group information.
* 5. Gets a list of Amazon Linux 2 AMIs and selects one.
* 6. Gets more information about the image.
* 7. Gets a list of instance types that are compatible with the selected AMI's
* architecture.
* 8. Creates an instance with the key pair, security group, AMI, and an
* instance type.
* 9. Displays information about the instance.
* 10. Stops the instance and waits for it to stop.
* 11. Starts the instance and waits for it to start.
* 12. Allocates an Elastic IP address and associates it with the instance.
* 13. Displays SSH connection info for the instance.
* 14. Disassociates and deletes the Elastic IP address.
* 15. Terminates the instance and waits for it to terminate.
* 16. Deletes the security group.
* 17. Deletes the key pair.
*/
public class EC2Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {

        final String usage = ""

            Usage:
            <keyName> <fileName> <groupName> <groupDesc> <vpcId>

            Where:
            keyName - A key pair name (for example, TestKeyPair).\s
            fileName - A file name where the key information is written to.
\s
            groupName - The name of the security group.\s
            groupDesc - The description of the security group.\s
            vpcId - A VPC Id value. You can get this value from the AWS
Management Console.\s
            myIpAddress - The IP address of your development machine.\s

            """;

        if (args.length != 6) {
            System.out.println(usage);

```

```
        System.exit(1);
    }

    String keyName = args[0];
    String fileName = args[1];
    String groupName = args[2];
    String groupDesc = args[3];
    String vpcId = args[4];
    String myIpAddress = args[5];

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

    SsmClient ssmClient = SsmClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon EC2 example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Create an RSA key pair and save the private key
material as a .pem file.");
    createKeyPair(ec2, keyName, fileName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. List key pairs.");
    describeKeys(ec2);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Create a security group.");
    String groupId = createSecurityGroup(ec2, groupName, groupDesc, vpcId,
myIpAddress);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Display security group info for the newly created
security group.");
    describeSecurityGroups(ec2, groupId);
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get a list of Amazon Linux 2 AMIs and selects one
with amzn2 in the name.");
String instanceId = getParaValues(ssmClient);
System.out.println("The instance Id is " + instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Get more information about an amzn2 image.");
String amiValue = describeImage(ec2, instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of instance types.");
String instanceType = getInstanceTypes(ec2);
System.out.println("The instance type is " + instanceType);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Create an instance.");
String newInstanceId = runInstance(ec2, instanceType, keyName, groupName,
amiValue);
System.out.println("The instance Id is " + newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Display information about the running instance. ");
String ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Stop the instance and use a waiter.");
stopInstance(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Start the instance and use a waiter.");
startInstance(ec2, newInstanceId);
ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
```

```
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Allocate an Elastic IP address and associate it with
the instance.");
String allocationId = allocateAddress(ec2);
System.out.println("The allocation Id value is " + allocationId);
String associationId = associateAddress(ec2, newInstanceId, allocationId);
System.out.println("The associate Id value is " + associationId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Describe the instance again.");
ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Disassociate and release the Elastic IP address.");
disassociateAddress(ec2, associationId);
releaseEC2Address(ec2, allocationId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Terminate the instance and use a waiter.");
terminateEC2(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("16. Delete the security group.");
deleteEC2SecGroup(ec2, groupId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("17. Delete the key.");
deleteKeys(ec2, keyName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("You successfully completed the Amazon EC2 scenario.");
System.out.println(DASHES);
ec2.close();
```

```
}

public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
    try {
        DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
        .groupId(groupId)
        .build();

        ec2.deleteSecurityGroup(request);
        System.out.println("Successfully deleted security group with Id " +
groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void terminateEC2(Ec2Client ec2, String instanceId) {
    try {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

        TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to
terminate. This will take a few minutes.");
        ec2.terminateInstances(ti);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
        .waitUntilInstanceTerminated(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
        System.out.println(instanceId + " is terminated!");
    }
}
```

```
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteKeys(Ec2Client ec2, String keyPair) {
        try {
            DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
                .keyName(keyPair)
                .build();

            ec2.deleteKeyPair(request);
            System.out.println("Successfully deleted key pair named " + keyPair);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void releaseEC2Address(Ec2Client ec2, String allocId) {
        try {
            ReleaseAddressRequest request = ReleaseAddressRequest.builder()
                .allocationId(allocId)
                .build();

            ec2.releaseAddress(request);
            System.out.println("Successfully released Elastic IP address " +
allocId);
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void disassociateAddress(Ec2Client ec2, String associationId) {
        try {
            DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
                .associationId(associationId)
                .build();

            ec2.disassociateAddress(addressRequest);
```

```
        System.out.println("You successfully disassociated the address!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
    try {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
        return associateResponse.associationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String allocateAddress(Ec2Client ec2) {
    try {
        AllocateAddressRequest allocateRequest =
AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        AllocateAddressResponse allocateResponse =
ec2.allocateAddress(allocateRequest);
        return allocateResponse.allocationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```



```
}

public static void startInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to run. This
will take a few minutes.");
    ec2.startInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully started instance " + instanceId);
}

public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();
}
```

```

        WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully stopped instance " + instanceId);
    }

    public static String describeEC2Instances(Ec2Client ec2, String newInstanceId) {
        try {
            String pubAddress = "";
            boolean isRunning = false;
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                .instanceIds(newInstanceId)
                .build();

            while (!isRunning) {
                DescribeInstancesResponse response = ec2.describeInstances(request);
                String state =
response.reservations().get(0).instances().get(0).state().name().name();
                if (state.compareTo("RUNNING") == 0) {
                    System.out.println("Image id is " +
response.reservations().get(0).instances().get(0).imageId());
                    System.out.println(
                        "Instance type is " +
response.reservations().get(0).instances().get(0).instanceType());
                    System.out.println(
                        "Instance state is " +
response.reservations().get(0).instances().get(0).state().name());
                    pubAddress =
response.reservations().get(0).instances().get(0).publicIpAddress();
                    System.out.println("Instance address is " + pubAddress);
                    isRunning = true;
                }
            }
            return pubAddress;
        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String runInstance(Ec2Client ec2, String instanceType, String
keyName, String groupName,
        String amiId) {

```

```

    try {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .instanceType(instanceType)
            .keyName(keyName)
            .securityGroups(groupName)
            .maxCount(1)
            .minCount(1)
            .imageId(amiId)
            .build();

        System.out.println("Going to start an EC2 instance using a waiter");
        RunInstancesResponse response = ec2.runInstances(runRequest);
        String instanceIdVal = response.instances().get(0).instanceId();
        ec2.waiter().waitUntilInstanceRunning(r ->
r.instanceIds(instanceIdVal));
        System.out.println("Successfully started EC2 instance " + instanceIdVal
+ " based on AMI " + amiId);
        return instanceIdVal;

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType;
    try {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .maxResults(10)
            .build();

        DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
        List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
        for (InstanceTypeInfo type : instanceTypes) {
            System.out.println("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
            System.out.println("Network information is " +
type.networkInfo().toString());

```

```
        System.out.println("Instance type is " +
type.instanceType().toString());
        instanceType = type.instanceType().toString();
        if (instanceType.compareTo("t2.2xlarge") == 0){
            return instanceType;
        }
    }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Display the Description field that corresponds to the instance Id value.
public static String describeImage(Ec2Client ec2, String instanceId) {
    try {
        DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
            .imageIds(instanceId)
            .build();

        DescribeImagesResponse response = ec2.describeImages(imagesRequest);
        System.out.println("The description of the first image is " +
response.images().get(0).description());
        System.out.println("The name of the first image is " +
response.images().get(0).name());

        // Return the image Id value.
        return response.images().get(0).imageId();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Get the Id value of an instance with amzn2 in the name.
public static String getParaValues(SsmClient ssmClient) {
    try {
        GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
            .path("/aws/service/ami-amazon-linux-latest")
```

```
        .build();

        GetParametersByPathIterable responses =
ssmClient.getParametersByPathPaginator(parameterRequest);
        for
(ssoftware.amazon.awssdk.services.ssm.model.GetParametersByPathResponse response :
responses) {
            System.out.println("Test " + response.nextToken());
            List<Parameter> parameterList = response.parameters();
            for (Parameter para : parameterList) {
                System.out.println("The name of the para is: " + para.name());
                System.out.println("The type of the para is: " + para.type());
                if (filterName(para.name())) {
                    return para.value();
                }
            }
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Return true if the name has amzn2 in it. For example:
// /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-arm64-gp2
private static boolean filterName(String name) {
    String[] parts = name.split("/");
    String myValue = parts[4];
    return myValue.contains("amzn2");
}

public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
```

```
        .flatMap(r -> r.securityGroups().stream())
        .forEach(group -> System.out
            .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
```

```

        .groupName(groupName)
        .ipPermissions(ipPerm, ipPerm2)
        .build();

    ec2.authorizeSecurityGroupIngress(authRequest);
    System.out.println("Successfully added ingress policy to security group
" + groupName);
    return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void describeKeys(Ec2Client ec2) {
    try {
        DescribeKeyPairsResponse response = ec2.describeKeyPairs();
        response.keyPairs().forEach(keyPair -> System.out.printf(
            "Found key pair with name %s " +
                "and fingerprint %s",
            keyPair.keyName(),
            keyPair.keyFingerprint()));

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createKeyPair(Ec2Client ec2, String keyName, String fileName)
{
    try {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();

        CreateKeyPairResponse response = ec2.createKeyPair(request);
        String content = response.keyMaterial();
        BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
        writer.write(content);
        writer.close();
        System.out.println("Successfully created key pair named " + keyName);
    }
}

```

```
        } catch (Ec2Exception | IOException e) {  
            System.err.println(e.getMessage());  
            System.exit(1);  
        }  
    }  
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [AllocateAddress](#)
  - [AssociateAddress](#)
  - [AuthorizeSecurityGroupIngress](#)
  - [CreateKeyPair](#)
  - [CreateSecurityGroup](#)
  - [DeleteKeyPair](#)
  - [DeleteSecurityGroup](#)
  - [DescribeImages](#)
  - [DescribeInstanceTypes](#)
  - [DescribeInstances](#)
  - [DescribeKeyPairs](#)
  - [DescribeSecurityGroups](#)
  - [DisassociateAddress](#)
  - [ReleaseAddress](#)
  - [RunInstances](#)
  - [StartInstances](#)
  - [StopInstances](#)
  - [TerminateInstances](#)
  - [UnmonitorInstances](#)

## Contoh Amazon ECS menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan kepada Anda cara melakukan tindakan dan menerapkan skenario umum AWS SDK for Java 2.x dengan menggunakan Amazon ECS.



Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

## Tindakan

Membuat kluster

Contoh kode berikut menunjukkan cara membuat cluster Amazon ECS.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandConfiguration;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandLogging;
import software.amazon.awssdk.services.ecs.model.ClusterConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateClusterResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.CreateClusterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateCluster {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <clusterName>\s

            Where:
                clusterName - The name of the ECS cluster to create.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        String clusterArn = createGivenCluster(ecsClient, clusterName);
        System.out.println("The cluster ARN is " + clusterArn);
        ecsClient.close();
    }

    public static String createGivenCluster(EcsClient ecsClient, String clusterName)
    {
        try {
            ExecuteCommandConfiguration commandConfiguration =
            ExecuteCommandConfiguration.builder()
                .logging(ExecuteCommandLogging.DEFAULT)
                .build();

            ClusterConfiguration clusterConfiguration =
            ClusterConfiguration.builder()
                .executeCommandConfiguration(commandConfiguration)
                .build();

            CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
```

```

        .clusterName(clusterName)
        .configuration(clusterConfiguration)
        .build();

    CreateClusterResponse response =
ecsClient.createCluster(clusterRequest);
    return response.cluster().clusterArn();

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}

```

- Untuk detail API, lihat [CreateCluster](#) di Referensi AWS SDK for Java 2.x API.

## Membuat layanan

Contoh kode berikut menunjukkan cara membuat layanan Amazon ECS.

### SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.AwsVpcConfiguration;
import software.amazon.awssdk.services.ecs.model.NetworkConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateServiceRequest;
import software.amazon.awssdk.services.ecs.model.LaunchType;
import software.amazon.awssdk.services.ecs.model.CreateServiceResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * Before running this Java V2 code example, set up your development

```

```

* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateService {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <clusterName> <serviceName> <securityGroups>
<subnets> <taskDefinition>

                Where:
                clusterName - The name of the ECS cluster.
                serviceName - The name of the ECS service to
create.

                securityGroups - The name of the security group.
                subnets - The name of the subnet.
                taskDefinition - The name of the task definition.
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceName = args[1];
        String securityGroups = args[2];
        String subnets = args[3];
        String taskDefinition = args[4];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        String serviceArn = createNewService(ecsClient, clusterName,
serviceName, securityGroups, subnets,
            taskDefinition);
        System.out.println("The ARN of the service is " + serviceArn);
        ecsClient.close();
    }
}

```

```
public static String createNewService(EcsClient ecsClient,
    String clusterName,
    String serviceName,
    String securityGroups,
    String subnets,
    String taskDefinition) {

    try {
        AwsVpcConfiguration vpcConfiguration =
        AwsVpcConfiguration.builder()
            .securityGroups(securityGroups)
            .subnets(subnets)
            .build();

        NetworkConfiguration configuration =
        NetworkConfiguration.builder()
            .awsvpcConfiguration(vpcConfiguration)
            .build();

        CreateServiceRequest serviceRequest =
        CreateServiceRequest.builder()
            .cluster(clusterName)
            .networkConfiguration(configuration)
            .desiredCount(1)
            .launchType(LaunchType.FARGATE)
            .serviceName(serviceName)
            .taskDefinition(taskDefinition)
            .build();

        CreateServiceResponse response =
        ecsClient.createService(serviceRequest);
        return response.service().serviceArn();

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Untuk detail API, lihat [CreateService](#) di Referensi AWS SDK for Java 2.x API.

## Hapus layanan

Contoh kode berikut menunjukkan cara menghapus layanan Amazon ECS.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DeleteServiceRequest;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteService {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <clusterName> <serviceArn>\s

            Where:
                clusterName - The name of the ECS cluster.
                serviceArn - The ARN of the ECS service.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String clusterName = args[0];
String serviceArn = args[1];
Region region = Region.US_EAST_1;
EcsClient ecsClient = EcsClient.builder()
    .region(region)
    .build();

deleteSpecificService(ecsClient, clusterName, serviceArn);
ecsClient.close();
}

public static void deleteSpecificService(EcsClient ecsClient, String
clusterName, String serviceArn) {
    try {
        DeleteServiceRequest serviceRequest = DeleteServiceRequest.builder()
            .cluster(clusterName)
            .service(serviceArn)
            .build();

        ecsClient.deleteService(serviceRequest);
        System.out.println("The Service was successfully deleted");

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DeleteService](#) di Referensi AWS SDK for Java 2.x API.

## Jelaskan cluster

Contoh kode berikut menunjukkan cara mendeskripsikan cluster Amazon ECS Anda.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeClustersRequest;
import software.amazon.awssdk.services.ecs.model.DescribeClustersResponse;
import software.amazon.awssdk.services.ecs.model.Cluster;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeClusters {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <clusterArn> \s

                Where:
                clusterArn - The ARN of the ECS cluster to describe.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterArn = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        descCluster(ecsClient, clusterArn);
    }

    public static void descCluster(EcsClient ecsClient, String clusterArn) {
        try {
```



```
DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
    .clusters(clusterArn)
    .build();

DescribeClustersResponse response =
ecsClient.describeClusters(clustersRequest);
List<Cluster> clusters = response.clusters();
for (Cluster cluster : clusters) {
    System.out.println("The cluster name is " + cluster.clusterName());
}

} catch (EcsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Untuk detail API, lihat [DescribeClusters](#) di Referensi AWS SDK for Java 2.x API.

## Jelaskan tugas

Contoh kode berikut menunjukkan cara mendeskripsikan tugas Amazon ECS Anda.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeTasksRequest;
import software.amazon.awssdk.services.ecs.model.DescribeTasksResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.Task;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTaskDefinitions {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <clusterArn> <taskId>\s

            Where:
                clusterArn - The ARN of an ECS cluster.
                taskId - The task Id value.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterArn = args[0];
        String taskId = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        getAllTasks(ecsClient, clusterArn, taskId);
        ecsClient.close();
    }

    public static void getAllTasks(EcsClient ecsClient, String clusterArn, String
taskId) {
        try {
            DescribeTasksRequest tasksRequest = DescribeTasksRequest.builder()
                .cluster(clusterArn)
                .tasks(taskId)
                .build();
```

```

        DescribeTasksResponse response = ecsClient.describeTasks(tasksRequest);
        List<Task> tasks = response.tasks();
        for (Task task : tasks) {
            System.out.println("The task ARN is " + task.taskDefinitionArn());
        }

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [DescribeTasks](#) di Referensi AWS SDK for Java 2.x API.

## Daftar kluster

Contoh kode berikut menunjukkan cara membuat daftar cluster Amazon ECS Anda.

### SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ListClustersResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

```

```
*/  
  
public class ListClusters {  
    public static void main(String[] args) {  
        Region region = Region.US_EAST_1;  
        EcsClient ecsClient = EcsClient.builder()  
            .region(region)  
            .build();  
  
        listAllClusters(ecsClient);  
        ecsClient.close();  
    }  
  
    public static void listAllClusters(EcsClient ecsClient) {  
        try {  
            ListClustersResponse response = ecsClient.listClusters();  
            List<String> clusters = response.clusterArns();  
            for (String cluster : clusters) {  
                System.out.println("The cluster arn is " + cluster);  
            }  
        } catch (EcsException e) {  
            System.err.println(e.awsErrorDetails().errorMessage());  
            System.exit(1);  
        }  
    }  
}
```

- Untuk detail API, lihat [ListClusters](#) di Referensi AWS SDK for Java 2.x API.

## Perbarui layanan

Contoh kode berikut menunjukkan cara memperbarui layanan Amazon ECS.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.UpdateServiceRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class UpdateService {

    public static void main(String[] args) {

        final String usage = ""

            Usage:
            <clusterName> <serviceArn>\s

            Where:
            clusterName - The cluster name.
            serviceArn - The service ARN value.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceArn = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        updateSpecificService(ecsClient, clusterName, serviceArn);
        ecsClient.close();
    }
}
```

```
public static void updateSpecificService(EcsClient ecsClient, String
clusterName, String serviceArn) {
    try {
        UpdateServiceRequest serviceRequest = UpdateServiceRequest.builder()
            .cluster(clusterName)
            .service(serviceArn)
            .desiredCount(0)
            .build();

        ecsClient.updateService(serviceRequest);
        System.out.println("The service was modified");

    } catch (EcsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [UpdateService](#) di Referensi AWS SDK for Java 2.x API.

## Contoh Elastic Load Balancing menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with Elastic Load Balancing.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Memulai

## Halo Elastic Load Balancing

Contoh kode berikut menunjukkan cara memulai menggunakan Elastic Load Balancing.

SDK for Java 2.x

### Note

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

```
public class HelloLoadBalancer {

    public static void main(String[] args) {
        ElasticLoadBalancingV2Client loadBalancingV2Client =
ElasticLoadBalancingV2Client.builder()
            .region(Region.US_EAST_1)
            .build();

        DescribeLoadBalancersResponse loadBalancersResponse =
loadBalancingV2Client
            .describeLoadBalancers(r -> r.pageSize(10));
        List<LoadBalancer> loadBalancerList =
loadBalancersResponse.loadBalancers();
        for (LoadBalancer lb : loadBalancerList)
            System.out.println("Load Balancer DNS name = " +
lb.dnsName());
    }
}
```

- Untuk detail API, lihat [DescribeLoadBalancers](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

Buat pendengar untuk menyeimbangkan beban

Contoh kode berikut menunjukkan cara membuat listener yang meneruskan permintaan dari penyeimbang beban ELB ke grup target.

SDK for Java 2.x

### Note

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

```
/*
 * Creates an Elastic Load Balancing load balancer that uses the specified
 * subnets
 * and forwards requests to the specified target group.
 */
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
```



```
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
        .loadBalancerArns(lbARN)
        .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
        .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
        .targetGroupArn(targetGroupARN)
        .type("forward")
        .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
        .defaultActions(action)
        .port(port)
        .protocol(protocol)
        .defaultActions(action)
        .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
        + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
```

```
}
```

- Untuk detail API, lihat [CreateListener](#) di Referensi AWS SDK for Java 2.x API.

## Buat grup target

Contoh kode berikut menunjukkan cara membuat grup target ELB.

## SDK for Java 2.x

### Note

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

```
/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
}
```

```
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }
}
```

- Untuk detail API, lihat [CreateTargetGroup](#) di Referensi AWS SDK for Java 2.x API.

## Membuat Application Load Balancer

Contoh kode berikut menunjukkan cara membuat ELB Application Load Balancer.

### SDK for Java 2.x

#### Note

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

```
/*
 * Creates an Elastic Load Balancing load balancer that uses the specified
 * subnets
 * and forwards requests to the specified target group.
 */
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
    }
}
```

```
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
```

```

        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}

```

- Untuk detail API, lihat [CreateLoadBalancer](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus penyeimbang beban

Contoh kode berikut menunjukkan cara menghapus penyeimbang beban ELB.

SDK for Java 2.x

### Note

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

```

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter

```

```

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

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

```

- Untuk detail API, lihat [DeleteLoadBalancer](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus grup target

Contoh kode berikut menunjukkan cara menghapus grup target ELB.

### SDK for Java 2.x

#### Note

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

```

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

```

- Untuk detail API, lihat [DeleteTargetGroup](#) di Referensi AWS SDK for Java 2.x API.

## Mendapatkan kesehatan kelompok sasaran

Contoh kode berikut menunjukkan cara mendapatkan kesehatan instance dalam kelompok target ELB.

SDK for Java 2.x

### Note

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

```
// Checks the health of the instances in the target group.
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
        .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}
```

- Untuk detail API, lihat [DescribeTargetHealth](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

### Membangun dan mengelola layanan yang tangguh

Contoh kode berikut menunjukkan cara membuat layanan web load-balanced yang mengembalikan rekomendasi buku, film, dan lagu. Contoh ini menunjukkan cara layanan tersebut merespons kegagalan, serta cara merestrukturisasi layanan agar lebih tangguh ketika terjadi kegagalan.

- Menggunakan grup Amazon EC2 Auto Scaling untuk membuat instans Amazon Elastic Compute Cloud (Amazon EC2) berdasarkan templat peluncuran dan menyimpan sejumlah instans dalam rentang yang ditentukan.
- Menangani dan mendistribusikan permintaan HTTP dengan Elastic Load Balancing.
- Memantau kondisi instans dalam grup Auto Scaling dan meneruskan permintaan hanya ke instans yang sehat.
- Menjalankan server web Python pada setiap instans EC2 untuk menangani permintaan HTTP. Server web merespons dengan memberikan rekomendasi dan melakukan pemeriksaan kondisi.
- Menyimulasikan layanan yang direkomendasikan dengan tabel Amazon DynamoDB.
- Kontrol respons server web terhadap permintaan dan pemeriksaan kesehatan dengan memperbarui AWS Systems Manager parameter.

### SDK for Java 2.x

#### Note

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

Menjalankan skenario interaktif di prompt perintah.

```
public class Main {  
  
    public static final String fileName = "C:\\AWS\\resworkflow\\  
\\recommendations.json"; // Modify file location.  
    public static final String tableName = "doc-example-recommendation-service";  
    public static final String startScript = "C:\\AWS\\resworkflow\\  
\\server_startup_script.sh"; // Modify file location.  
}
```



```
    public static final String policyFile = "C:\\\\AWS\\\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\\\AWS\\\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
    {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
    }
}
```

```

System.out.println(DASHES);
System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
System.out.println("Press Enter when you're ready.");
in.nextLine();
demo(loadBalancer);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("C - DELETE THE RESOURCES");
System.out.println("""
    This concludes the demo of how to build and manage a resilient
service.

    To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
    that were created for this demo.
    """);

System.out.println("\n Do you want to delete the resources (y/n)? ");
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

if (userInput.equals("y")) {
    // Delete resources here
    deleteResources(loadBalancer, autoScaler, database);
    System.out.println("Resources deleted.");
} else {
    System.out.println("""
        Okay, we'll leave the resources intact.
        Don't forget to delete them when you're done with them or you
might incur unexpected charges.
        """);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The example has completed. ");
System.out.println("\n Thanks for watching!");
System.out.println(DASHES);
}

// Deletes the AWS resources used in this example.
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
    throws IOException, InterruptedException {
    loadBalancer.deleteLoadBalancer(lbName);

```

```

        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScalingGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
    InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
            """
                For this demo, we'll use the AWS SDK for Java (v2) to create
    several AWS resources
                to set up a load-balanced web service endpoint and explore
    some ways to make it resilient
                against various kinds of failures.

                Some of the resources create by this demo are:
                \t* A DynamoDB table that the web service depends on to
    provide book, movie, and song recommendations.
                \t* An EC2 launch template that defines EC2 instances that
    each contain a Python web server.
                \t* An EC2 Auto Scaling group that manages EC2 instances
    across several Availability Zones.
                \t* An Elastic Load Balancing (ELB) load balancer that
    targets the Auto Scaling group to distribute requests.
            """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
        tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""

```

Creating an EC2 launch template that runs '{startup\_script}' when an instance starts.

This script starts a Python web server defined in the `server.py` script. The web server

listens to HTTP requests on port 80 and responds to requests to '/' and to '/healthcheck'.

For demo purposes, this server is run as the root user. In production, the best practice is to run a web server, such as Apache, with least-privileged credentials.

The template also defines an IAM policy that each instance uses to assume a role that grants permissions to access the DynamoDB recommendation table and Systems Manager parameters that control the flow of the demo.

```
""");
```

```
LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println(
    "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
System.out.println("*** Wait 30 secs for the VPC to be created");
TimeUnit.SECONDS.sleep(30);
AutoScaler autoScaler = new AutoScaler();
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);
```

```
System.out.println("""
    At this point, you have EC2 instances created. Once each instance
starts, it listens for
    HTTP requests. You can see these instances in the console or
continue with the demo.
    Press Enter when you're ready to continue.
""");
```

```
in.nextLine();
System.out.println(DASHES);
```

```
System.out.println(DASHES);
```

```
System.out.println("Creating variables that control the flow of the demo.");
ParameterHelper paramHelper = new ParameterHelper();
paramHelper.reset();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an Elastic Load Balancing target group and load balancer.
The target group
    defines how the load balancer connects to instances. The load
balancer provides a
    single endpoint where clients connect and dispatches requests to
instances in the group.
    """);

String vpcId = autoScaler.getDefaultVPC();
List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
System.out.println("Verifying access to the load balancer endpoint...");
boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
if (!wasSuccessful) {
    System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to "http://checkip.amazonaws.com"
   HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
    try {
        // Execute the request and get the response
        HttpResponse response = httpClient.execute(httpGet);

        // Read the response content.
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

        // Print the public IP address.
        System.out.println("Public IP Address: " + ipAddress);
```

```
        GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
        if (!groupInfo.isPortOpen()) {
            System.out.println("""
                For this example to work, the default security group for
your default VPC must
                allow access from this computer. You can either add it
automatically from this
                example or add it yourself using the AWS Management
Console.
                """);

            System.out.println(
                "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
            System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
            String ans = in.nextLine();
            if ("y".equalsIgnoreCase(ans)) {
                autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                System.out.println("Security group rule added.");
            } else {
                System.out.println("No security group rule added.");
            }
        }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
} else if (wasSuccessful) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" + elbDnsName);
} else {
    System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
    System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
    System.out.println("you can successfully make a GET request to the load
balancer.");
}
```

```

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
            """
                This part of the demonstration shows how to toggle
different parts of the system
                to create situations where the web service fails, and shows
how using a resilient
                architecture can keep the web service running in spite of
these failures.

                At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
            """);
        demoChoices(loadBalancer);

        System.out.println(
            """
                The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
            """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
            """
                \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as

```

```

        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
        """);
demoChoices(loadBalancer);

System.out.println(
    ""
        Instead of failing when the recommendation service fails,
the web service can return a static response.
        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
        """);
paramHelper.put(paramHelper.failureResponse, "static");

System.out.println("""
    Now, sending a GET request to the load balancer endpoint returns a
static response.
    The service still reports as healthy because health checks are still
shallow.
        """);
demoChoices(loadBalancer);

System.out.println("Let's reinstate the recommendation service.");
paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

System.out.println("""
    Let's also substitute bad credentials for one of the instances in
the target group so that it can't
    access the DynamoDB recommendation table. We will get an instance id
value.
        """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
AutoScaler autoScaler = new AutoScaler();

// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " + profileAssociationId);

```



```
System.out.println("Replacing the profile for instance " + badInstanceId
    + " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
    ""
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
    "");

demoChoices(loadBalancer);

System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
        the web service can access the DynamoDB table that it depends on for
recommendations. Note that
        the deep health check is only for ELB routing and not for Auto
Scaling instance health.
    This kind of deep health check is not recommended for Auto Scaling
instance health, because it
        risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
    """);

System.out.println("""
    By implementing deep health checks, the load balancer can detect
when one of the instances is failing
        and take that instance out of rotation.
    """);

paramHelper.put(paramHelper.healthCheck, "deep");

System.out.println("""
    Now, checking target health indicates that the instance with bad
credentials
        is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
        instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
        the load balancer takes unhealthy instances out of its rotation.
    """);
```

```

        """);

demoChoices(loadBalancer);

System.out.println(
    ""
        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
        instance is to terminate it and let the auto scaler start a
new instance to replace it.
        """);
autoScaler.terminateInstance(badInstanceId);

System.out.println("""
    Even while the instance is terminating and the new instance is
starting, sending a GET
    request to the web service continues to get a successful
recommendation response because
    the load balancer routes requests to the healthy instances. After
the replacement instance
    starts and reports as healthy, it is included in the load balancing
rotation.

    Note that terminating and replacing an instance typically takes
several minutes, during which time you
    can see the changing health check status until the new instance is
running and healthy.
    """);

demoChoices(loadBalancer);
System.out.println(
    "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

demoChoices(loadBalancer);
paramHelper.reset();
}

public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    String[] actions = {
        "Send a GET request to the load balancer endpoint.",
        "Check the health of load balancer targets.",
        "Go to the next part of the demo."
    }

```

```
};
Scanner scanner = new Scanner(System.in);

while (true) {
    System.out.println("-".repeat(88));
    System.out.println("See the current state of the service by selecting
one of the following choices:");
    for (int i = 0; i < actions.length; i++) {
        System.out.println(i + ": " + actions[i]);
    }

    try {
        System.out.print("\nWhich action would you like to take? ");
        int choice = scanner.nextInt();
        System.out.println("-".repeat(88));

        switch (choice) {
            case 0 -> {
                System.out.println("Request:\n");
                System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                CloseableHttpClient httpClient =
HttpClients.createDefault();

                // Create an HTTP GET request to the ELB.
                HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);

                // Display the JSON response
                BufferedReader reader = new BufferedReader(
                    new
InputStreamReader(response.getEntity().getContent()));
                StringBuilder jsonResponse = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null) {
                    jsonResponse.append(line);
                }
                reader.close();
            }
        }
    }
}
```

```

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();

    }
    case 1 -> {
        System.out.println("\nChecking the health of load balancer
targets:\n");

        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
        Note that it can take a minute or two for the health
check to update
        after changes are made.
        """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value between
0-2. Please select again.");
}

} catch (java.util.InputMismatchException e) {
    System.out.println("Invalid input. Please select again.");
    scanner.nextLine(); // Clear the input buffer.
}
}
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}

```

```
}  
}
```

Membuat kelas yang menggabungkan tindakan Penskalaan Otomatis dan Amazon EC2.

```
public class AutoScaler {  
  
    private static Ec2Client ec2Client;  
    private static AutoScalingClient autoScalingClient;  
    private static IamClient iamClient;  
  
    private static SsmClient ssmClient;  
  
    private IamClient getIAMClient() {  
        if (iamClient == null) {  
            iamClient = IamClient.builder()  
                .region(Region.US_EAST_1)  
                .build();  
        }  
        return iamClient;  
    }  
  
    private SsmClient getSSMClient() {  
        if (ssmClient == null) {  
            ssmClient = SsmClient.builder()  
                .region(Region.US_EAST_1)  
                .build();  
        }  
        return ssmClient;  
    }  
  
    private Ec2Client getEc2Client() {  
        if (ec2Client == null) {  
            ec2Client = Ec2Client.builder()  
                .region(Region.US_EAST_1)  
                .build();  
        }  
        return ec2Client;  
    }  
  
    private AutoScalingClient getAutoScalingClient() {  
        if (autoScalingClient == null) {
```

```
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
    software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
        .builder()
        .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                // name.
        .build();
}
```

```
// Replace the IAM instance profile association for the EC2 instance.
ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
    .builder()
    .iamInstanceProfile(iamInstanceProfile)
    .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
    .build();

try {
    getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
    // Handle the response as needed.
} catch (Ec2Exception e) {
    // Handle exceptions, log, or report the error.
    System.err.println("Error: " + e.getMessage());
}

System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
    newInstanceProfileName);
TimeUnit.SECONDS.sleep(15);
boolean instReady = false;
int tries = 0;

// Reboot after 60 seconds
while (!instReady) {
    if (tries % 6 == 0) {
        getEc2Client().rebootInstances(RebootInstancesRequest.builder()
            .instanceIds(instanceId)
            .build());
        System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
    }
    tries++;
    try {
        TimeUnit.SECONDS.sleep(10);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
    List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
    for (InstanceInformation info : instanceInformationList) {
```

```

        if (info.instanceId().equals(instanceId)) {
            instReady = true;
            break;
        }
    }
}

SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
    .instanceIds(instanceId)
    .documentName("AWS-RunShellScript")
    .parameters(Collections.singletonMap("commands",
        Collections.singletonList("cd / && sudo python3 server.py
80")))
    .build();

getSSMClient().sendCommand(sendCommandRequest);
System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
}

public void openInboundPort(String secGroupId, String port, String ipAddress) {
    AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(secGroupId)
        .cidrIp(ipAddress)
        .fromPort(Integer.parseInt(port))
        .build();

    getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
    System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
}

/**
 * Detaches a role from an instance profile, detaches policies from the role,
 * and deletes all the resources.
 */
public void deleteInstanceProfile(String roleName, String profileName) {
    try {
        software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
        .builder()
        .instanceProfileName(profileName)

```



```
        .build();

        GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
        String name = response.getInstanceProfile().getInstanceProfileName();
        System.out.println(name);

        RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .roleName(roleName)
            .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
            .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(attachedPolicy.policyArn())
                .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");
```

```
        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /**
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                .name("group-name")
                .values("default")

```

```
        .build());

    Filter filter1 = Filter.builder()
        .name("vpc-id")
        .values(VPC)
        .build();

    DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
        .filters(filter, filter1)
        .build();

    DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
        .describeSecurityGroups(securityGroupsRequest);
    String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
    groupInfo.setGroupName(securityGroup);

    for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
        System.out.println("Found security group: " + secGroup.groupId());

        for (IpPermission ipPermission : secGroup.ipPermissions()) {
            if (ipPermission.fromPort() == port) {
                System.out.println("Found inbound rule: " + ipPermission);
                for (IpRange ipRange : ipPermission.ipRanges()) {
                    String cidrIp = ipRange.cidrIp();
                    if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                        System.out.println(cidrIp + " is applicable");
                        portIsOpen = true;
                    }
                }

                if (!ipPermission.prefixListIds().isEmpty()) {
                    System.out.println("Prefix lList is applicable");
                    portIsOpen = true;
                }

                if (!portIsOpen) {
                    System.out
                        .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                }
            }
        }
    }
}
```

```

        } else {
            break;
        }
    }
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/*
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

```

```
// Get availability zones.
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
    .builder()
    .build();

DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

String availabilityZones = String.join(",", availabilityZoneNames);
LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

String[] zones = availabilityZones.split(",");
CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

try {
    getAutoScalingClient().createAutoScalingGroup(groupRequest);

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
return zones;
}
```

```
public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();

    DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
        .filters(vpcFilter, azFilter, defaultForAZ)
        .build();

    DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
    subnets = response.subnets();
    return subnets;
}
```

```
// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
```

```

        ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                .builder()
                .policyArn(policy.arn())
                .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .roleName(roleName) // Specify the name of the IAM role
                        .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
                    .policyArn(policy.arn())
                    .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

// List the roles associated with the instance profile

```



```

        ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
    .roleName(roleName) // Remove the extra dot here
    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
    .build();

        getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}

```

Membuat kelas yang menggabungkan tindakan Penyeimbangan Beban Elastis.

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()

```

```
        .region(Region.US_EAST_1)
        .build();
    }

    return elasticLoadBalancingV2Client;
}

// Checks the health of the instances in the target group.
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
        .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
```

```

        .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
        .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {

```

```
        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);
        if (statusCode == 200) {
            success = true;
        } else {
            retries--;
            System.out.println("Got connection error from load balancer
endpoint, retrying...");
            TimeUnit.SECONDS.sleep(15);
        }
    }

} catch (org.apache.http.conn.HttpHostConnectException e) {
    System.out.println(e.getMessage());
}

System.out.println("Status.." + success);
return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
}
```

```

        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
                .scheme("internet-facing")
                .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                .loadBalancerArns(lbARN)
                .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                .waitUntilLoadBalancerAvailable(request);

```

```

waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Load Balancer " + lbName + " is available.");

// Get the DNS name (endpoint) of the load balancer.
String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

// Create a listener for the load balance.
Action action = Action.builder()
    .targetGroupArn(targetGroupARN)
    .type("forward")
    .build();

CreateListenerRequest listenerRequest = CreateListenerRequest.builder()
    .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
    .defaultActions(action)
    .port(port)
    .protocol(protocol)
    .defaultActions(action)
    .build();

getLoadBalancerClient().createListener(listenerRequest);
System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
    + targetGroupARN);

// Return the load balancer DNS name.
return lbDNSName;

} catch (ElasticLoadBalancingV2Exception e) {
    e.printStackTrace();
}
return "";
}
}

```

Membuat kelas yang menggunakan DynamoDB untuk menyimulasikan layanan yang direkomendasikan.

```
public class Database {
```

```
private static DynamoDbClient dynamoDbClient;

public static DynamoDbClient getDynamoDbClient() {
    if (dynamoDbClient == null) {
        dynamoDbClient = DynamoDbClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return dynamoDbClient;
}

// Checks to see if the Amazon DynamoDB table exists.
private boolean doesTableExist(String tableName) {
    try {
        // Describe the table and catch any exceptions.
        DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        getDynamoDbClient().describeTable(describeTableRequest);
        System.out.println("Table '" + tableName + "' exists.");
        return true;
    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " + e.getMessage());
    }
    return false;
}

/**
 * Creates a DynamoDB table to use a recommendation service. The table has a
 * hash key named 'MediaType' that defines the type of media recommended, such
 * as
 * Book or Movie, and a range key named 'ItemId' that, combined with the
 * MediaType,
 * forms a unique identifier for the recommended item.
 */
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
```

```
DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
CreateTableRequest createTableRequest = CreateTableRequest.builder()
    .tableName(tableName)
    .attributeDefinitions(
        AttributeDefinition.builder()
            .attributeName("MediaType")
            .attributeType(ScalarAttributeType.S)
            .build(),
        AttributeDefinition.builder()
            .attributeName("ItemId")
            .attributeType(ScalarAttributeType.N)
            .build()
    )
    .keySchema(
        KeySchemaElement.builder()
            .attributeName("MediaType")
            .keyType(KeyType.HASH)
            .build(),
        KeySchemaElement.builder()
            .attributeName("ItemId")
            .keyType(KeyType.RANGE)
            .build()
    )
    .provisionedThroughput(
        ProvisionedThroughput.builder()
            .readCapacityUnits(5L)
            .writeCapacityUnits(5L)
            .build()
    )
    .build();

getDynamoDbClient().createTable(createTableRequest);
System.out.println("Creating table " + tableName + "...");

// Wait until the Amazon DynamoDB table is created.
DescribeTableRequest tableRequest = DescribeTableRequest.builder()
    .tableName(tableName)
    .build();

WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Table " + tableName + " created.");

// Add records to the table.
populateTable(fileName, tableName);
}
```



```
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
    System.out.println("Added all records to the " + tableName);
}
}
```

Membuat kelas yang menggabungkan tindakan Systems Manager.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
  - [AttachLoadBalancerTargetGroups](#)
  - [CreateAutoScalingGroup](#)
  - [CreateInstanceProfile](#)
  - [CreateLaunchTemplate](#)
  - [CreateListener](#)
  - [CreateLoadBalancer](#)
  - [CreateTargetGroup](#)
  - [DeleteAutoScalingGroup](#)

- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacelamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

## MediaStore contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with MediaStore.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

- [Tindakan](#)

## Tindakan

### Buat wadah

Contoh kode berikut menunjukkan cara membuat AWS Elemental MediaStore wadah.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <containerName>

            Where:
                containerName - The name of the container to create.
            """;
    }
}
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }

    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
        String containerName) {
        try {
            CreateContainerRequest containerRequest =
                CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
                mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
                    containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
                containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [CreateContainer](#) di Referensi AWS SDK for Java 2.x API.

## Hapus wadah

Contoh kode berikut menunjukkan cara menghapus AWS Elemental MediaStore wadah.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <containerName>

            Where:
                containerName - The name of the container to create.
        """;
    }
}
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }

    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
        String containerName) {
        try {
            CreateContainerRequest containerRequest =
                CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
                mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
                    containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
                containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DeleteContainer](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus objek

Contoh kode berikut menunjukkan cara menghapus AWS Elemental MediaStore objek.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.DeleteObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

                Usage:    <completePath> <containerName>

                Where:
```



```
        completePath - The path (including the container) of the item to
delete.
        containerName - The name of the container.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String completePath = args[0];
    String containerName = args[1];
    Region region = Region.US_EAST_1;
    URI uri = new URI(getEndpoint(containerName));

    MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
        .endpointOverride(uri)
        .region(region)
        .build();

    deleteMediaObject(mediaStoreData, completePath);
    mediaStoreData.close();
}

public static void deleteMediaObject(MediaStoreDataClient mediaStoreData, String
completePath) {
    try {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
            .path(completePath)
            .build();

        mediaStoreData.deleteObject(deleteObjectRequest);

    } catch (MediaStoreDataException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

private static String getEndpoint(String containerName) {
    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();
```

```
        DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
        .containerName(containerName)
        .build();


        DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
        mediaStoreClient.close();
        return response.container().endpoint();
    }
}
```

- Untuk detail API, lihat [DeleteObject](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan sebuah wadah

Contoh kode berikut menunjukkan bagaimana mendeskripsikan sebuah AWS Elemental MediaStore wadah.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
*/
public class DescribeContainer {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <containerName>

            Where:
                containerName - The name of the container to describe.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        System.out.println("Status is " + checkContainer(mediaStoreClient,
            containerName));
        mediaStoreClient.close();
    }

    public static String checkContainer(MediaStoreClient mediaStoreClient, String
        containerName) {
        try {
            DescribeContainerRequest describeContainerRequest =
            DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

            DescribeContainerResponse containerResponse =
            mediaStoreClient.describeContainer(describeContainerRequest);
            System.out.println("The container name is " +
            containerResponse.container().name());
            System.out.println("The container ARN is " +
            containerResponse.container().arn());
            return containerResponse.container().status().toString();
        }
    }
}
```

```
    } catch (MediaStoreException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Untuk detail API, lihat [DescribeContainer](#) di Referensi AWS SDK for Java 2.x API.

## Dapatkan objek

Contoh kode berikut menunjukkan bagaimana untuk mendapatkan AWS Elemental MediaStore objek.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectResponse;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class GetObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

            Usage:    <completePath> <containerName> <savePath>

            Where:
                completePath - The path of the object in the container (for
example, Videos5/sampleVideo.mp4).
                containerName - The name of the container.
                savePath - The path on the local drive where the file is saved,
including the file name (for example, C:/AWS/myvid.mp4).
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        String savePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
            .endpointOverride(uri)
            .region(region)
            .build();

        getMediaObject(mediaStoreData, completePath, savePath);
        mediaStoreData.close();
    }

    public static void getMediaObject(MediaStoreDataClient mediaStoreData, String
completePath, String savePath) {

        try {
```

```
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .path(completePath)
            .build();

        // Write out the data to a file.
        ResponseInputStream<GetObjectResponse> data =
mediaStoreData.getObject(objectRequest);
        byte[] buffer = new byte[data.available()];
        data.read(buffer);

        File targetFile = new File(savePath);
        OutputStream outputStream = new FileOutputStream(targetFile);
        outputStream.write(buffer);
        System.out.println("The data was written to " + savePath);

    } catch (MediaStoreDataException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

private static String getEndpoint(String containerName) {
    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();

    DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
        .containerName(containerName)
        .build();

    DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
    return response.container().endpoint();
}
}
```

- Untuk detail API, lihat [GetObject](#) di Referensi AWS SDK for Java 2.x API.

## Daftar kontainer

Contoh kode berikut menunjukkan bagaimana untuk daftar AWS Elemental MediaStore kontainer.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.Container;
import software.amazon.awssdk.services.mediastore.model.ListContainersResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListContainers {

    public static void main(String[] args) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        listAllContainers(mediaStoreClient);
        mediaStoreClient.close();
    }

    public static void listAllContainers(MediaStoreClient mediaStoreClient) {
        try {
```

```
        ListContainersResponse containersResponse =
mediaStoreClient.listContainers();
        List<Container> containers = containersResponse.containers();
        for (Container container : containers) {
            System.out.println("Container name is " + container.name());
        }

    } catch (MediaStoreException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [ListContainers](#) di Referensi AWS SDK for Java 2.x API.

Masukkan benda ke dalam wadah

Contoh kode berikut menunjukkan bagaimana menempatkan objek ke dalam AWS Elemental MediaStore wadah.

SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectResponse;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import java.io.File;
import java.net.URI;
import java.net.URISyntaxException;
```



```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObject {
    public static void main(String[] args) throws URISyntaxException {
        final String USAGE = ""

            To run this example, supply the name of a container, a file location
            to use, and path in the container\s

                Ex: <containerName> <filePath> <completePath>
                """;

        if (args.length < 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String containerName = args[0];
        String filePath = args[1];
        String completePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
            .endpointOverride(uri)
            .region(region)
            .build();

        putMediaObject(mediaStoreData, filePath, completePath);
        mediaStoreData.close();
    }

    public static void putMediaObject(MediaStoreDataClient mediaStoreData, String
filePath, String completePath) {
        try {
            File myFile = new File(filePath);
            RequestBody requestBody = RequestBody.fromFile(myFile);
```

```
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .path(completePath)
            .contentType("video/mp4")
            .build();

        PutObjectResponse response = mediaStoreData.putObject(objectRequest,
requestBody);
        System.out.println("The saved object is " +
response.storageClass().toString());

    } catch (MediaStoreDataException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String getEndpoint(String containerName) {

    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();

    DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
        .containerName(containerName)
        .build();

    DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
    return response.container().endpoint();
}
}
```

- Untuk detail API, lihat [PutObject](#) di Referensi AWS SDK for Java 2.x API.

## OpenSearch Contoh layanan menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan kepada Anda cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan OpenSearch Layanan AWS SDK for Java 2.x with.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

## Tindakan

Membuat domain

Contoh kode berikut menunjukkan cara membuat domain OpenSearch Service.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.VolumeType;
import software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.CreateDomainResponse;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateDomain {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <domainName>

            Where:
                domainName - The name of the domain to create.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .build();

        createNewDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void createNewDomain(OpenSearchClient searchClient, String
domainName) {
        try {
            ClusterConfig clusterConfig = ClusterConfig.builder()
                .dedicatedMasterEnabled(true)
                .dedicatedMasterCount(3)
                .dedicatedMasterType("t2.small.search")
                .instanceType("t2.small.search")
                .instanceCount(5)
                .build();

            EBSOptions ebsOptions = EBSOptions.builder()
                .ebsEnabled(true)
                .volumeSize(10)
```

```
        .volumeType(VolumeType.GP2)
        .build();

        NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
        .enabled(true)
        .build();

        CreateDomainRequest domainRequest = CreateDomainRequest.builder()
        .domainName(domainName)
        .engineVersion("OpenSearch_1.0")
        .clusterConfig(clusterConfig)
        .ebsOptions(ebsOptions)
        .nodeToNodeEncryptionOptions(encryptionOptions)
        .build();

        System.out.println("Sending domain creation request...");
        CreateDomainResponse createResponse =
searchClient.createDomain(domainRequest);
        System.out.println("Domain status is " +
createResponse.domainStatus().toString());
        System.out.println("Domain Id is " +
createResponse.domainStatus().domainId());

        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [CreateDomain](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus domain

Contoh kode berikut menunjukkan cara menghapus domain OpenSearch Layanan.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDomain {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <domainName>

                Where:
                domainName - The name of the domain to delete.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
                .region(region)
                .build();
```

```
        deleteSpecificDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void deleteSpecificDomain(OpenSearchClient searchClient, String
domainName) {
        try {
            DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
                .domainName(domainName)
                .build();

            searchClient.deleteDomain(domainRequest);
            System.out.println(domainName + " was successfully deleted.");

        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DeleteDomain](#) di Referensi AWS SDK for Java 2.x API.

## Daftar domain

Contoh kode berikut menunjukkan cara membuat daftar domain OpenSearch Layanan.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.DomainInfo;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesRequest;
```

```
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesResponse;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListDomainNames {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();
        listAllDomains(searchClient);
        System.out.println("Done");
    }

    public static void listAllDomains(OpenSearchClient searchClient) {
        try {
            ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
                .engineType("OpenSearch")
                .build();

            ListDomainNamesResponse response =
searchClient.listDomainNames(namesRequest);
            List<DomainInfo> domainInfoList = response.domainNames();
            for (DomainInfo domain : domainInfoList)
                System.out.println("Domain name is " + domain.domainName());

        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```



- Untuk detail API, lihat [ListDomainNames](#) di Referensi AWS SDK for Java 2.x API.

## Memodifikasi konfigurasi cluster

Contoh kode berikut menunjukkan cara memodifikasi konfigurasi cluster domain OpenSearch Service.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateDomain {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <domainName>

                Where:
                domainName - The name of the domain to update.

        """;
```

```

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .build();

        updateSpecificDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void updateSpecificDomain(OpenSearchClient searchClient, String
domainName) {
        try {
            ClusterConfig clusterConfig = ClusterConfig.builder()
                .instanceCount(3)
                .build();

            UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
                .domainName(domainName)
                .clusterConfig(clusterConfig)
                .build();

            System.out.println("Sending domain update request...");
            UpdateDomainConfigResponse updateResponse =
searchClient.updateDomainConfig(updateDomainConfigRequest);
            System.out.println("Domain update response from Amazon OpenSearch
Service:");
            System.out.println(updateResponse.toString());

        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Untuk detail API, lihat [UpdateDomainConfig](#) di Referensi AWS SDK for Java 2.x API.

## EventBridge contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with EventBridge.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Memulai

Halo EventBridge

Contoh kode berikut menunjukkan cara untuk mulai menggunakan EventBridge.

SDK for Java 2.x

### Note

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

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */
public class HelloEventBridge {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
```

```
EventBridgeClient eventBrClient = EventBridgeClient.builder()
    .region(region)
    .build();

listBuses(eventBrClient);
eventBrClient.close();
}

public static void listBuses(EventBridgeClient eventBrClient) {
    try {
        ListEventBusesRequest busesRequest = ListEventBusesRequest.builder()
            .limit(10)
            .build();

        ListEventBusesResponse response =
eventBrClient.listEventBuses(busesRequest);
        List<EventBus> buses = response.eventBuses();
        for (EventBus bus : buses) {
            System.out.println("The name of the event bus is: " + bus.name());
            System.out.println("The ARN of the event bus is: " + bus.arn());
        }

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [ListEventBuses](#) di Referensi AWS SDK for Java 2.x API.

## Topik


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

## Tindakan

### Tambahkan target

Contoh kode berikut menunjukkan cara menambahkan target ke EventBridge acara Amazon.

## SDK for Java 2.x

 Note

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

Tambahkan topik Amazon SNS sebagai target aturan.

```
// Add a rule which triggers an SNS target when a file is uploaded to an S3
// bucket.
public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
    String topicName, String eventRuleName, String bucketName) {
    String targetID = java.util.UUID.randomUUID().toString();
    Target myTarget = Target.builder()
        .id(targetID)
        .arn(topicArn)
        .build();

    List<Target> targets = new ArrayList<>();
    targets.add(myTarget);
    PutTargetsRequest request = PutTargetsRequest.builder()
        .eventBusName(null)
        .targets(targets)
        .rule(ruleName)
        .build();

    eventBrClient.putTargets(request);
    System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
target " + topicName + " for bucket "
        + bucketName + ".");
}
```

Tambahkan transformator input ke target untuk aturan.

```
public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
    String ruleName) {
    String targetId = java.util.UUID.randomUUID().toString();
```

```
InputTransformer inputTransformer = InputTransformer.builder()
    .inputTemplate("\Notification: sample event was received.\")
    .build();

Target target = Target.builder()
    .id(targetId)
    .arn(topicArn)
    .inputTransformer(inputTransformer)
    .build();

try {
    PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
        .rule(ruleName)
        .targets(target)
        .eventBusName(null)
        .build();

    eventBrClient.putTargets(targetsRequest);
} catch (EventBridgeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Untuk detail API, lihat [PutTargets](#) di Referensi AWS SDK for Java 2.x API.

## Buat aturan

Contoh kode berikut menunjukkan cara membuat EventBridge aturan Amazon.

## SDK for Java 2.x

### Note

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

Buat aturan terjadwal.

```

public static void createEBRule(EventBridgeClient eventBrClient, String
ruleName, String cronExpression) {
    try {
        PutRuleRequest ruleRequest = PutRuleRequest.builder()
            .name(ruleName)
            .eventBusName("default")
            .scheduleExpression(cronExpression)
            .state("ENABLED")
            .description("A test rule that runs on a schedule created by the
Java API")
            .build();

        PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
        System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

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.
public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
    String eventRuleName) {
    String pattern = "{\n" +
        "  \"source\": [\"aws.s3\"],\n" +
        "  \"detail-type\": [\"Object Created\"],\n" +
        "  \"detail\": {\n" +
        "    \"bucket\": {\n" +
        "      \"name\": [\"" + bucketName + "\"] \n" +
        "    } \n" +
        "  } \n" +
        "}";

    try {
        PutRuleRequest ruleRequest = PutRuleRequest.builder()
            .description("Created by using the AWS SDK for Java v2")

```

```
        .name(eventRuleName)
        .eventPattern(pattern)
        .roleArn(roleArn)
        .build();

        PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
        System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [PutRule](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus peraturan

Contoh kode berikut menunjukkan cara menghapus EventBridge aturan Amazon.

SDK for Java 2.x

### Note

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

```
public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
    DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
        .name(ruleName)
        .build();

    eventBrClient.deleteRule(ruleRequest);
    System.out.println("Successfully deleted the rule");
}
```




- Untuk detail API, lihat [DeleteRule](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan aturan

Contoh kode berikut menunjukkan cara mendeskripsikan EventBridge aturan Amazon.

SDK for Java 2.x

 Note

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

```
public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    try {
        DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
            .name(eventRuleName)
            .build();

        DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
        System.out.println("The state of the rule is " +
response.stateAsString());


    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DescribeRule](#) di Referensi AWS SDK for Java 2.x API.

Nonaktifkan aturan

Contoh kode berikut menunjukkan cara menonaktifkan EventBridge aturan Amazon.

## SDK for Java 2.x

 Note

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

Nonaktifkan aturan dengan menggunakan nama aturannya.

```
public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
    try {
        if (!isEnabled) {
            System.out.println("Disabling the rule: " + eventRuleName);
            DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                .name(eventRuleName)
                .build();


            eventBrClient.disableRule(ruleRequest);
        } else {
            System.out.println("Enabling the rule: " + eventRuleName);
            EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                .name(eventRuleName)
                .build();
            eventBrClient.enableRule(ruleRequest);
        }
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DisableRule](#) di Referensi AWS SDK for Java 2.x API.

Aktifkan aturan

Contoh kode berikut menunjukkan cara mengaktifkan EventBridge aturan Amazon.

## SDK for Java 2.x

 Note

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

Aktifkan aturan dengan menggunakan nama aturannya.

```
public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
    try {
        if (!isEnabled) {
            System.out.println("Disabling the rule: " + eventRuleName);
            DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                .name(eventRuleName)
                .build();

            eventBrClient.disableRule(ruleRequest);
        } else {
            System.out.println("Enabling the rule: " + eventRuleName);
            EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                .name(eventRuleName)
                .build();
            eventBrClient.enableRule(ruleRequest);
        }
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [EnableRule](#) di Referensi AWS SDK for Java 2.x API.

Daftar nama aturan untuk target

Contoh kode berikut menunjukkan cara mencantumkan nama EventBridge aturan Amazon untuk target.

## SDK for Java 2.x

 Note

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

Buat daftar semua nama aturan dengan menggunakan target.

```
public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
    ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
        .targetArn(topicArn)
        .build();


    ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
    List<String> rules = response.ruleNames();
    for (String rule : rules) {
        System.out.println("The rule name is " + rule);
    }
}
```

- Untuk detail API, lihat [ListRuleNamesByTarget](#) di Referensi AWS SDK for Java 2.x API.

## Daftar aturan

Contoh kode berikut menunjukkan cara membuat daftar EventBridge aturan Amazon.

## SDK for Java 2.x

 Note

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

Aktifkan aturan dengan menggunakan nama aturannya.

```
public static void listRules(EventBridgeClient eventBrClient) {
    try {
        ListRulesRequest rulesRequest = ListRulesRequest.builder()
            .eventBusName("default")
            .limit(10)
            .build();

        ListRulesResponse response = eventBrClient.listRules(rulesRequest);
        List<Rule> rules = response.rules();
        for (Rule rule : rules) {
            System.out.println("The rule name is : " + rule.name());
            System.out.println("The rule description is : " +
rule.description());
            System.out.println("The rule state is : " + rule.stateAsString());
        }

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ListRules](#) di Referensi AWS SDK for Java 2.x API.

## Daftar target untuk aturan

Contoh kode berikut menunjukkan cara mencantumkan EventBridge target Amazon untuk aturan.

## SDK for Java 2.x

### Note

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

Buat daftar semua target untuk aturan dengan menggunakan nama aturan.

```
public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
{
```

```
ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
    .rule(ruleName)
    .build();

ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
List<Target> targetsList = res.targets();
for (Target target: targetsList) {
    System.out.println("Target ARN: "+target.arn());
}
}
```

- Untuk detail API, lihat [ListTargetsByRule](#) di Referensi AWS SDK for Java 2.x API.

## Hapus target dari aturan

Contoh kode berikut menunjukkan cara menghapus EventBridge target Amazon dari aturan.

### SDK for Java 2.x

#### Note

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

Hapus semua target untuk aturan dengan menggunakan nama aturan.

```
public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    // First, get all targets that will be deleted.
    ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
        .rule(eventRuleName)
        .build();

    ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
    List<Target> allTargets = response.targets();

    // Get all targets and delete them.
    for (Target myTarget : allTargets) {
```

```
RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
    .rule(eventRuleName)
    .ids(myTarget.id())
    .build();

eventBrClient.removeTargets(removeTargetsRequest);
System.out.println("Successfully removed the target");
}
}
```

- Untuk detail API, lihat [RemoveTargets](#) di Referensi AWS SDK for Java 2.x API.

## Kirim acara

Contoh kode berikut menunjukkan cara mengirim EventBridge acara Amazon.

## SDK for Java 2.x

### Note

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

```
public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
    String json = "{" +
        "\"UserEmail\": \"" + email + "\", " +
        "\"Message\": \"This event was generated by example code.\", " +
        "\"UtcTime\": \"Now.\" " +
        "}";

    PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
        .source("ExampleSource")
        .detail(json)
        .detailType("ExampleType")
        .build();

    PutEventsRequest eventsRequest = PutEventsRequest.builder()
        .entries(entry)
```

```
        .build();

    eventBridgeClient.putEvents(eventsRequest);
}
```

- Untuk detail API, lihat [PutEvents](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

Memulai dengan aturan dan target

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat aturan dan tambahkan target ke dalamnya.
- Aktifkan dan nonaktifkan aturan.
- Daftar dan perbarui aturan dan target.
- Kirim acara, lalu bersihkan sumber daya.

SDK for Java 2.x

### Note

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

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * This Java V2 example performs the following tasks with Amazon EventBridge:
 *
 * 1. Creates an AWS Identity and Access Management (IAM) role to use with
```



```

* Amazon EventBridge.
* 2. Amazon Simple Storage Service (Amazon S3) bucket with EventBridge events
* enabled.
* 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
* 4. Lists rules on the event bus.
* 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and
* lets the user subscribe to it.
* 6. Adds a target to the rule that sends an email to the specified topic.
* 7. Creates an EventBridge event that sends an email when an Amazon S3 object
* is created.
* 8. Lists Targets.
* 9. Lists the rules for the same target.
* 10. Triggers the rule by uploading a file to the Amazon S3 bucket.
* 11. Disables a specific rule.
* 12. Checks and print the state of the rule.
* 13. Adds a transform to the rule to change the text of the email.
* 14. Enables a specific rule.
* 15. Triggers the updated rule by uploading a file to the Amazon S3 bucket.
* 16. Updates the rule to be a custom rule pattern.
* 17. Sending an event to trigger the rule.
* 18. Cleans up resources.
*
*/
public class EventbridgeMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException, IOException
    {
        final String usage = ""

            Usage:
                <roleName> <bucketName> <topicName> <eventRuleName>

            Where:
                roleName - The name of the role to create.
                bucketName - The Amazon Simple Storage Service (Amazon S3)
bucket name to create.
                topicName - The name of the Amazon Simple Notification Service
(Amazon SNS) topic to create.
                eventRuleName - The Amazon EventBridge rule name to create.
            """;

        if (args.length != 5) {
            System.out.println(usage);

```

```
        System.exit(1);
    }

    String polJSON = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": \"events.amazonaws.com\"" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]}" +
        "}";

    Scanner sc = new Scanner(System.in);
    String roleName = args[0];
    String bucketName = args[1];
    String topicName = args[2];
    String eventRuleName = args[3];

    Region region = Region.US_EAST_1;
    EventBridgeClient eventBrClient = EventBridgeClient.builder()
        .region(region)
        .build();

    S3Client s3Client = S3Client.builder()
        .region(region)
        .build();

    Region regionGl = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(regionGl)
        .build();

    SnsClient snsClient = SnsClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon EventBridge example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out
```

```
        .println("1. Create an AWS Identity and Access Management (IAM) role
to use with Amazon EventBridge.");
        String roleArn = createIAMRole(iam, roleName, polJSON);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create an S3 bucket with EventBridge events
enabled.");
        if (checkBucket(s3Client, bucketName)) {
            System.out.println("Bucket " + bucketName + " already exists. Ending
this scenario.");
            System.exit(1);
        }

        createBucket(s3Client, bucketName);
        Thread.sleep(3000);
        setBucketNotification(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a rule that triggers when an object is
uploaded to Amazon S3.");
        Thread.sleep(10000);
        addEventRule(eventBrClient, roleArn, bucketName, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. List rules on the event bus.");
        listRules(eventBrClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create a new SNS topic for testing and let the user
subscribe to the topic.");
        String topicArn = createSnsTopic(snsClient, topicName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add a target to the rule that sends an email to the
specified topic.");
        System.out.println("Enter your email to subscribe to the Amazon SNS
topic:");
        String email = sc.nextLine();
        subEmail(snsClient, topicArn, email);
```

```
System.out.println(
    "Use the link in the email you received to confirm your
subscription. Then, press Enter to continue.");
sc.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Create an EventBridge event that sends an email when
an Amazon S3 object is created.");
addSnsEventRule(eventBrClient, eventRuleName, topicArn, topicName,
eventRuleName, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 8. List Targets.");
listTargets(eventBrClient, eventRuleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 9. List the rules for the same target.");
listTargetRules(eventBrClient, topicArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 10. Trigger the rule by uploading a file to the S3
bucket.");
System.out.println("Press Enter to continue.");
sc.nextLine();
uploadTextFiletoS3(s3Client, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Disable a specific rule.");
changeRuleState(eventBrClient, eventRuleName, false);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Check and print the state of the rule.");
checkRule(eventBrClient, eventRuleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Add a transform to the rule to change the text of
the email.");
```

```
updateSnsEventRule(eventBrClient, topicArn, eventRuleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Enable a specific rule.");
changeRuleState(eventBrClient, eventRuleName, true);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 15. Trigger the updated rule by uploading a file to the
S3 bucket.");
System.out.println("Press Enter to continue.");
sc.nextLine();
uploadTextFiletoS3(s3Client, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 16. Update the rule to be a custom rule pattern.");
updateToCustomRule(eventBrClient, eventRuleName);
System.out.println("Updated event rule " + eventRuleName + " to use a custom
pattern.");
updateCustomRuleTargetWithTransform(eventBrClient, topicArn, eventRuleName);
System.out.println("Updated event target " + topicArn + ".");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("17. Sending an event to trigger the rule. This will
trigger a subscription email.");
triggerCustomRule(eventBrClient, email);
System.out.println("Events have been sent. Press Enter to continue.");
sc.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("18. Clean up resources.");
System.out.println("Do you want to clean up resources (y/n)");
String ans = sc.nextLine();
if (ans.compareTo("y") == 0) {
    cleanupResources(eventBrClient, snsClient, s3Client, iam, topicArn,
eventRuleName, bucketName, roleName);
} else {
    System.out.println("The resources will not be cleaned up. ");
}
System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("The Amazon EventBridge example scenario has successfully
completed.");
        System.out.println(DASHES);
    }

    public static void cleanupResources(EventBridgeClient eventBrClient, SnsClient
snsClient, S3Client s3Client,
        IamClient iam, String topicArn, String eventRuleName, String bucketName,
String roleName) {
        System.out.println("Removing all targets from the event rule.");
        deleteTargetsFromRule(eventBrClient, eventRuleName);
        deleteRuleByName(eventBrClient, eventRuleName);
        deleteSNSTopic(snsClient, topicArn);
        deleteS3Bucket(s3Client, bucketName);
        deleteRole(iam, roleName);
    }

    public static void deleteRole(IamClient iam, String roleName) {
        String policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess";
        DetachRolePolicyRequest policyRequest = DetachRolePolicyRequest.builder()
            .policyArn(policyArn)
            .roleName(roleName)
            .build();

        iam.detachRolePolicy(policyRequest);
        System.out.println("Successfully detached policy " + policyArn + " from role
" + roleName);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);
    }

    public static void deleteS3Bucket(S3Client s3Client, String bucketName) {
        // Remove all the objects from the S3 bucket.
        ListObjectsRequest listObjects = ListObjectsRequest.builder()
            .bucket(bucketName)
            .build();
```

```
ListObjectsResponse res = s3Client.listObjects(listObjects);
List<S3Object> objects = res.contents();
ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();

for (S3Object myValue : objects) {
    toDelete.add(ObjectIdentifier.builder()
        .key(myValue.key())
        .build());
}

DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
    .bucket(bucketName)
    .delete(Delete.builder()
        .objects(toDelete).build())
    .build();

s3Client.deleteObjects(dor);

// Delete the S3 bucket.
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucketName)
    .build();

s3Client.deleteBucket(deleteBucketRequest);
System.out.println("You have deleted the bucket and the objects");
}

// Delete the SNS topic.
public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
    try {
        DeleteTopicRequest request = DeleteTopicRequest.builder()
            .topicArn(topicArn)
            .build();

        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
    DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
        .name(ruleName)
        .build();

    eventBrClient.deleteRule(ruleRequest);
    System.out.println("Successfully deleted the rule");
}

public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    // First, get all targets that will be deleted.
    ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
        .rule(eventRuleName)
        .build();

    ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
    List<Target> allTargets = response.targets();

    // Get all targets and delete them.
    for (Target myTarget : allTargets) {
        RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
            .rule(eventRuleName)
            .ids(myTarget.id())
            .build();

        eventBrClient.removeTargets(removeTargetsRequest);
        System.out.println("Successfully removed the target");
    }
}

public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
    String json = "{" +
        "\"UserEmail\": \"" + email + "\", " +
        "\"Message\": \"This event was generated by example code.\", " +
        "\"UtcTime\": \"Now.\"";
    }";

    PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
```



```

        .source("ExampleSource")
        .detail(json)
        .detailType("ExampleType")
        .build();

    PutEventsRequest eventsRequest = PutEventsRequest.builder()
        .entries(entry)
        .build();

    eventBrClient.putEvents(eventsRequest);
}

public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
    String ruleName) {
    String targetId = java.util.UUID.randomUUID().toString();
    InputTransformer inputTransformer = InputTransformer.builder()
        .inputTemplate("\Notification: sample event was received.\")")
        .build();

    Target target = Target.builder()
        .id(targetId)
        .arn(topicArn)
        .inputTransformer(inputTransformer)
        .build();

    try {
        PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
            .rule(ruleName)
            .targets(target)
            .eventBusName(null)
            .build();

        eventBrClient.putTargets(targetsRequest);
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateToCustomRule(EventBridgeClient eventBrClient, String
ruleName) {
    String customEventsPattern = "{" +
        "\"source\": [\"ExampleSource\"],\" +

```

```
        "\"detail-type\": [\"ExampleType\"]" +
        "});

    PutRuleRequest request = PutRuleRequest.builder()
        .name(ruleName)
        .description("Custom test rule")
        .eventPattern(customEventsPattern)
        .build();

    eventBrClient.putRule(request);
}

// Update an Amazon S3 object created rule with a transform on the target.
public static void updateSnsEventRule(EventBridgeClient eventBrClient, String
topicArn, String ruleName) {
    String targetId = java.util.UUID.randomUUID().toString();
    Map<String, String> myMap = new HashMap<>();
    myMap.put("bucket", "$.detail.bucket.name");
    myMap.put("time", "$.time");

    InputTransformer inputTransformer = InputTransformer.builder()
        .inputTemplate("\"Notification: an object was uploaded to bucket
<bucket> at <time>.\")")
        .inputPathsMap(myMap)
        .build();

    Target target = Target.builder()
        .id(targetId)
        .arn(topicArn)
        .inputTransformer(inputTransformer)
        .build();

    try {
        PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
            .rule(ruleName)
            .targets(target)
            .eventBusName(null)
            .build();

        eventBrClient.putTargets(targetsRequest);

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
  
  public static void checkRule(EventBridgeClient eventBrClient, String  
eventRuleName) {  
    try {  
      DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()  
        .name(eventRuleName)  
        .build();  
  
      DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);  
      System.out.println("The state of the rule is " +  
response.stateAsString());  
  
    } catch (EventBridgeException e) {  
      System.err.println(e.awsErrorDetails().errorMessage());  
      System.exit(1);  
    }  
  }  
  
  public static void changeRuleState(EventBridgeClient eventBrClient, String  
eventRuleName, Boolean isEnabled) {  
    try {  
      if (!isEnabled) {  
        System.out.println("Disabling the rule: " + eventRuleName);  
        DisableRuleRequest ruleRequest = DisableRuleRequest.builder()  
          .name(eventRuleName)  
          .build();  
  
        eventBrClient.disableRule(ruleRequest);  
      } else {  
        System.out.println("Enabling the rule: " + eventRuleName);  
        EnableRuleRequest ruleRequest = EnableRuleRequest.builder()  
          .name(eventRuleName)  
          .build();  
        eventBrClient.enableRule(ruleRequest);  
      }  
  
    } catch (EventBridgeException e) {  
      System.err.println(e.awsErrorDetails().errorMessage());  
      System.exit(1);  
    }  
  }  
}
```

```
// Create and upload a file to an S3 bucket to trigger an event.
public static void uploadTextFiletoS3(S3Client s3Client, String bucketName)
throws IOException {
    // Create a unique file name.
    String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new
Date());
    String fileName = "TextFile" + fileSuffix + ".txt";

    File myFile = new File(fileName);
    FileWriter fw = new FileWriter(myFile.getAbsolutePath());
    BufferedWriter bw = new BufferedWriter(fw);
    bw.write("This is a sample file for testing uploads.");
    bw.close();

    try {
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(fileName)
            .build();

        s3Client.putObject(putOb, RequestBody.fromFile(myFile));

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
    ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
        .targetArn(topicArn)
        .build();

    ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
    List<String> rules = response.ruleNames();
    for (String rule : rules) {
        System.out.println("The rule name is " + rule);
    }
}
```

```

    public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
    {
        ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
            .rule(ruleName)
            .build();

        ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
        List<Target> targetsList = res.targets();
        for (Target target: targetsList) {
            System.out.println("Target ARN: "+target.arn());
        }
    }

    // Add a rule which triggers an SNS target when a file is uploaded to an S3
    // bucket.
    public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
        String topicName, String eventRuleName, String bucketName) {
        String targetID = java.util.UUID.randomUUID().toString();
        Target myTarget = Target.builder()
            .id(targetID)
            .arn(topicArn)
            .build();

        List<Target> targets = new ArrayList<>();
        targets.add(myTarget);
        PutTargetsRequest request = PutTargetsRequest.builder()
            .eventBusName(null)
            .targets(targets)
            .rule(ruleName)
            .build();

        eventBrClient.putTargets(request);
        System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
target " + topicName + " for bucket "
            + bucketName + ".");
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String email)
    {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("email")

```

```

        .endpoint(email)
        .returnSubscriptionArn(true)
        .topicArn(topicArn)
        .build();

    SubscribeResponse result = snsClient.subscribe(request);
    System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
        + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void listRules(EventBridgeClient eventBrClient) {
    try {
        ListRulesRequest rulesRequest = ListRulesRequest.builder()
            .eventBusName("default")
            .limit(10)
            .build();

        ListRulesResponse response = eventBrClient.listRules(rulesRequest);
        List<Rule> rules = response.rules();
        for (Rule rule : rules) {
            System.out.println("The rule name is : " + rule.name());
            System.out.println("The rule description is : " +
rule.description());
            System.out.println("The rule state is : " + rule.stateAsString());
        }

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String createSnsTopic(SnsClient snsClient, String topicName) {
    String topicPolicy = "{" +
        "\"Version\": \"2012-10-17\", " +
        "\"Statement\": [{" +
        "\"Sid\": \"EventBridgePublishTopic\", " +
        "\"Effect\": \"Allow\", " +

```

```

        "\"Principal\": {" +
        "\"Service\": \"events.amazonaws.com\"" +
        "}," +
        "\"Resource\": \"*\",\" +
        "\"Action\": \"sns:Publish\"" +
        "}]\" +
        "}";

    Map<String, String> topicAttributes = new HashMap<>();
    topicAttributes.put("Policy", topicPolicy);
    CreateTopicRequest topicRequest = CreateTopicRequest.builder()
        .name(topicName)
        .attributes(topicAttributes)
        .build();

    CreateTopicResponse response = snsClient.createTopic(topicRequest);
    System.out.println("Added topic " + topicName + " for email
subscriptions.");
    return response.topicArn();
}

// Create a new event rule that triggers when an Amazon S3 object is created in
// a bucket.
public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
    String eventRuleName) {
    String pattern = "{\n" +
        "  \"source\": [\"aws.s3\"],\n" +
        "  \"detail-type\": [\"Object Created\"],\n" +
        "  \"detail\": {\n" +
        "    \"bucket\": {\n" +
        "      \"name\": [\"\" + bucketName + "\"]\n" +
        "    }\n" +
        "  }\n" +
        "};

    try {
        PutRuleRequest ruleRequest = PutRuleRequest.builder()
            .description("Created by using the AWS SDK for Java v2")
            .name(eventRuleName)
            .eventPattern(pattern)
            .roleArn(roleArn)
            .build();

```

```
        PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
        System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Determine if the S3 bucket exists.
public static Boolean checkBucket(S3Client s3Client, String bucketName) {
    try {
        HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.headBucket(headBucketRequest);
        return true;
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    return false;
}

// Set the S3 bucket notification configuration.
public static void setBucketNotification(S3Client s3Client, String bucketName) {
    try {
        EventBridgeConfiguration eventBridgeConfiguration =
EventBridgeConfiguration.builder()
            .build();

        NotificationConfiguration configuration =
NotificationConfiguration.builder()
            .eventBridgeConfiguration(eventBridgeConfiguration)
            .build();

        PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
            .builder()
            .bucket(bucketName)
            .notificationConfiguration(configuration)
            .skipDestinationValidation(true)
            .build();
```



```
        s3Client.putBucketNotificationConfiguration(configurationRequest);
        System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(polJSON)
            .description("Created using the AWS SDK for Java")
            .build();
```

```
        CreateRoleResponse response = iam.createRole(request);
        AttachRolePolicyRequest rolePolicyRequest =
AttachRolePolicyRequest.builder()
            .roleName(rolename)
            .policyArn("arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess")
            .build();

        iam.attachRolePolicy(rolePolicyRequest);
        return response.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [DeleteRule](#)
  - [DescribeRule](#)
  - [DisableRule](#)
  - [EnableRule](#)
  - [ListRuleNamesByTarget](#)
  - [ListRules](#)
  - [ListTargetsByRule](#)
  - [PutEvents](#)
  - [PutRule](#)
  - [PutTargets](#)

## Contoh Forecast menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with Forecast.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

## Tindakan

Buat kumpulan data

Contoh kode berikut menunjukkan cara membuat kumpulan data Forecast.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateDatasetRequest;
import software.amazon.awssdk.services.forecast.model.Schema;
import software.amazon.awssdk.services.forecast.model.SchemaAttribute;
import software.amazon.awssdk.services.forecast.model.CreateDatasetResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```

* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateDataSet {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <name>\s

            Where:
                name - The name of the data set.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        String myDataSetARN = createForecastDataSet(forecast, name);
        System.out.println("The ARN of the new data set is " + myDataSetARN);
        forecast.close();
    }

    public static String createForecastDataSet(ForecastClient forecast, String name)
    {
        try {
            Schema schema = Schema.builder()
                .attributes(getSchema())
                .build();

            CreateDatasetRequest datasetRequest = CreateDatasetRequest.builder()
                .datasetName(name)
                .domain("CUSTOM")
                .datasetType("RELATED_TIME_SERIES")
                .dataFrequency("D")
                .schema(schema)

```

```
        .build();

        CreateDatasetResponse response = forecast.createDataset(datasetRequest);
        return response.datasetArn();

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

// Create a SchemaAttribute list required to create a data set.
private static List<SchemaAttribute> getSchema() {

    List<SchemaAttribute> schemaList = new ArrayList<>();
    SchemaAttribute att1 = SchemaAttribute.builder()
        .attributeName("item_id")
        .attributeType("string")
        .build();

    SchemaAttribute att2 = SchemaAttribute.builder()
        .attributeName("timestamp")
        .attributeType("timestamp")
        .build();

    SchemaAttribute att3 = SchemaAttribute.builder()
        .attributeName("target_value")
        .attributeType("float")
        .build();

    // Push the SchemaAttribute objects to the List.
    schemaList.add(att1);
    schemaList.add(att2);
    schemaList.add(att3);
    return schemaList;
}
}
```

- Untuk detail API, lihat [CreateDataset](#) di Referensi AWS SDK for Java 2.x API.

## Buat ramalan

Contoh kode berikut menunjukkan cara membuat perkiraan Forecast.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateForecastRequest;
import software.amazon.awssdk.services.forecast.model.CreateForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateForecast {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <name> <predictorArn>\s

            Where:
                name - The name of the forecast.\s
                predictorArn - The arn of the predictor to use.\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String name = args[0];
String predictorArn = args[1];
Region region = Region.US_WEST_2;
ForecastClient forecast = ForecastClient.builder()
    .region(region)
    .build();

String forecastArn = createNewForecast(forecast, name, predictorArn);
System.out.println("The ARN of the new forecast is " + forecastArn);
forecast.close();
}

public static String createNewForecast(ForecastClient forecast, String name,
String predictorArn) {
    try {
        CreateForecastRequest forecastRequest = CreateForecastRequest.builder()
            .forecastName(name)
            .predictorArn(predictorArn)
            .build();

        CreateForecastResponse response =
forecast.createForecast(forecastRequest);
        return response.forecastArn();

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Untuk detail API, lihat [CreateForecast](#) di Referensi AWS SDK for Java 2.x API.

## Hapus kumpulan data

Contoh kode berikut menunjukkan cara menghapus kumpulan data Forecast.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataset {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <datasetARN>\s

            Where:
                datasetARN - The ARN of the data set to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String datasetARN = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
```



```
        .build();

        deleteForecastDataSet(forecast, datasetARN);
        forecast.close();
    }

    public static void deleteForecastDataSet(ForecastClient forecast, String
myDataSetARN) {
        try {
            DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
                .datasetArn(myDataSetARN)
                .build();

            forecast.deleteDataset(deleteRequest);
            System.out.println("The Data Set was deleted");

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DeleteDataset](#) di Referensi AWS SDK for Java 2.x API.

## Hapus perkiraan

Contoh kode berikut menunjukkan cara menghapus perkiraan Forecast.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataset {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <datasetARN>\s

            Where:
                datasetARN - The ARN of the data set to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String datasetARN = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        deleteForecastDataSet(forecast, datasetARN);
        forecast.close();
    }

    public static void deleteForecastDataSet(ForecastClient forecast, String
myDataSetARN) {
        try {
            DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
                .datasetArn(myDataSetARN)
                .build();

            forecast.deleteDataset(deleteRequest);
        }
    }
}
```

```

        System.out.println("The Data Set was deleted");

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- Untuk detail API, lihat [DeleteForecast](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan ramalan

Contoh kode berikut menunjukkan bagaimana menggambarkan perkiraan Forecast.

SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DescribeForecastRequest;
import software.amazon.awssdk.services.forecast.model.DescribeForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeForecast {
    public static void main(String[] args) {
        final String usage = ""

```

```

        Usage:
            <forecastarn>\s

        Where:
            forecastarn - The arn of the forecast (for example,
"arn:aws:forecast:us-west-2:xxxxx322:forecast/my_forecast)
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String forecastarn = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        describe(forecast, forecastarn);
        forecast.close();
    }

    public static void describe(ForecastClient forecast, String forecastarn) {
        try {
            DescribeForecastRequest request = DescribeForecastRequest.builder()
                .forecastArn(forecastarn)
                .build();

            DescribeForecastResponse response = forecast.describeForecast(request);
            System.out.println("The name of the forecast is " +
response.forecastName());

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Untuk detail API, lihat [DescribeForecast](#) di Referensi AWS SDK for Java 2.x API.

## Daftar kelompok kumpulan data

Contoh kode berikut menunjukkan cara daftar grup kumpulan data Forecast.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DatasetGroupSummary;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsRequest;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListDataSetGroups {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        listDataGroups(forecast);
        forecast.close();
    }

    public static void listDataGroups(ForecastClient forecast) {
        try {
            ListDatasetGroupsRequest group = ListDatasetGroupsRequest.builder()
                .maxResults(10)
```

```

        .build();

        ListDatasetGroupsResponse response = forecast.listDatasetGroups(group);
        List<DatasetGroupSummary> groups = response.datasetGroups();
        for (DatasetGroupSummary myGroup : groups) {
            System.out.println("The Data Set name is " +
myGroup.datasetGroupName());
        }

    } catch (ForecastException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- Untuk detail API, lihat [ListDatasetGroups](#) di Referensi AWS SDK for Java 2.x API.

## Daftar prakiraan

Contoh kode berikut menunjukkan cara membuat daftar perkiraan Forecast.

## SDK for Java 2.x

### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.ListForecastsResponse;
import software.amazon.awssdk.services.forecast.model.ListForecastsRequest;
import software.amazon.awssdk.services.forecast.model.ForecastSummary;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development

```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListForecasts {

    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        listAllForecasts(forecast);
        forecast.close();
    }

    public static void listAllForecasts(ForecastClient forecast) {
        try {
            ListForecastsRequest request = ListForecastsRequest.builder()
                .maxResults(10)
                .build();

            ListForecastsResponse response = forecast.listForecasts(request);
            List<ForecastSummary> forecasts = response.forecasts();
            for (ForecastSummary forecastSummary : forecasts) {
                System.out.println("The name of the forecast is " +
forecastSummary.forecastName());
            }

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListForecasts](#) di Referensi AWS SDK for Java 2.x API.

## AWS Glue contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with AWS Glue.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Memulai

Halo AWS Glue

Contoh kode berikut menunjukkan cara untuk mulai menggunakan AWS Glue.

SDK for Java 2.x

### Note

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

```
package com.example.glue;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.ListJobsRequest;
import software.amazon.awssdk.services.glue.model.ListJobsResponse;
import java.util.List;

public class HelloGlue {
    public static void main(String[] args) {
        GlueClient glueClient = GlueClient.builder()
            .region(Region.US_EAST_1)
            .build();
```



```
        listJobs(glueClient);
    }

    public static void listJobs(GlueClient glueClient) {
        ListJobsRequest request = ListJobsRequest.builder()
            .maxResults(10)
            .build();
        ListJobsResponse response = glueClient.listJobs(request);
        List<String> jobList = response.jobNames();
        jobList.forEach(job -> {
            System.out.println("Job Name: " + job);
        });
    }
}
```

- Untuk detail API, lihat [ListJobs](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

### Buat crawler

Contoh kode berikut menunjukkan cara membuat AWS Glue crawler.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.CreateCrawlerRequest;
```

```
import software.amazon.awssdk.services.glue.model.CrawlerTargets;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.S3Target;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCrawler {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <IAM> <s3Path> <cron> <dbName> <crawlerName>

            Where:
                IAM - The ARN of the IAM role that has AWS Glue and S3
permissions.\s
                s3Path - The Amazon Simple Storage Service (Amazon S3) target
that contains data (for example, CSV data).
                cron - A cron expression used to specify the schedule (i.e.,
cron(15 12 * * ? *).
                dbName - The database name.\s
                crawlerName - The name of the crawler.\s
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
```

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

    createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
    glueClient.close();
}

public static void createGlueCrawler(GlueClient glueClient,
    String iam,
    String s3Path,
    String cron,
    String dbName,
    String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)
            .build();

        // Add the S3Target to a list.
        List<S3Target> targetList = new ArrayList<>();
        targetList.add(s3Target);

        CrawlerTargets targets = CrawlerTargets.builder()
            .s3Targets(targetList)
            .build();

        CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
            .databaseName(dbName)
            .name(crawlerName)
            .description("Created by the AWS Glue Java API")
            .targets(targets)
            .role(iam)
            .schedule(cron)
            .build();

        glueClient.createCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully created");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- Untuk detail API, lihat [CreateCrawler](#) di Referensi AWS SDK for Java 2.x API.

## Dapatkan crawler

Contoh kode berikut menunjukkan cara mendapatkan AWS Glue crawler.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetCrawlerRequest;
import software.amazon.awssdk.services.glue.model.GetCrawlerResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetCrawler {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <crawlerName>
```

```
        Where:
            crawlerName - The name of the crawler.\s
            """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String crawlerName = args[0];
    Region region = Region.US_EAST_1;
    GlueClient glueClient = GlueClient.builder()
        .region(region)
        .build();

    getSpecificCrawler(glueClient, crawlerName);
    glueClient.close();
}

public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
{
    try {
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
        Instant createDate = response.crawler().creationTime();

        // Convert the Instant to readable date
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the Crawler is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- Untuk detail API, lihat [GetCrawler](#) di Referensi AWS SDK for Java 2.x API.

## Dapatkan database dari Katalog Data

Contoh kode berikut menunjukkan bagaimana untuk mendapatkan database dari AWS Glue Data Catalog.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetDatabaseRequest;
import software.amazon.awssdk.services.glue.model.GetDatabaseResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetDatabase {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
```

```
        <databaseName>

        Where:
            databaseName - The name of the database.\s
            """";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String databaseName = args[0];
    Region region = Region.US_EAST_1;
    GlueClient glueClient = GlueClient.builder()
        .region(region)
        .build();

    getSpecificDatabase(glueClient, databaseName);
    glueClient.close();
}

public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
    try {
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
            .name(databaseName)
            .build();

        GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
        Instant createDate = response.database().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the database is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```


```
}  
}
```

- Untuk detail API, lihat [GetDatabase](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan tabel dari database

Contoh kode berikut menunjukkan bagaimana untuk mendapatkan tabel dari database di AWS Glue Data Catalog.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.glue.GlueClient;  
import software.amazon.awssdk.services.glue.model.GetTableRequest;  
import software.amazon.awssdk.services.glue.model.GetTableResponse;  
import software.amazon.awssdk.services.glue.model.GlueException;  
import java.time.Instant;  
import java.time.ZoneId;  
import java.time.format.DateTimeFormatter;  
import java.time.format.FormatStyle;  
import java.util.Locale;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class GetTable {  
    public static void main(String[] args) {  
        final String usage = ""
```



```
Usage:
    <dbName> <tableName>

Where:
    dbName - The database name.\s
    tableName - The name of the table.\s
""";

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String dbName = args[0];
String tableName = args[1];
Region region = Region.US_EAST_1;
GlueClient glueClient = GlueClient.builder()
    .region(region)
    .build();

getGlueTable(glueClient, dbName, tableName);
glueClient.close();
}

public static void getGlueTable(GlueClient glueClient, String dbName, String
tableName) {
    try {
        GetTableRequest tableRequest = GetTableRequest.builder()
            .databaseName(dbName)
            .name(tableName)
            .build();

        GetTableResponse tableResponse = glueClient.getTable(tableRequest);
        Instant createDate = tableResponse.table().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the table is " + createDate);
    }
}
```

```
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [GetTables](#) di Referensi AWS SDK for Java 2.x API.

## Mulai crawler

Contoh kode berikut menunjukkan cara memulai AWS Glue crawler.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.StartCrawlerRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StartCrawler {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <crawlerName>
```

```

        Where:
            crawlerName - The name of the crawler.\s
            """";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String crawlerName = args[0];
    Region region = Region.US_EAST_1;
    GlueClient glueClient = GlueClient.builder()
        .region(region)
        .build();

    startSpecificCrawler(glueClient, crawlerName);
    glueClient.close();
}

public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
    try {
        StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.startCrawler(crawlerRequest);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [StartCrawler](#) di Referensi AWS SDK for Java 2.x API.

## Skenario


Memulai crawler dan lowongan kerja

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat crawler yang merayapi bucket Amazon S3 publik dan membuat database metadata berformat CSV.
- Buat 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 for Java 2.x

 Note

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

```
/**
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To set up the resources, see this documentation topic:
 *
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
 *
 * This example performs the following tasks:
 *
 * 1. Create a database.
 * 2. Create a crawler.
 * 3. Get a crawler.
 * 4. Start a crawler.
 * 5. Get a database.
 * 6. Get tables.
 * 7. Create a job.
```

```

* 8. Start a job run.
* 9. List all jobs.
* 10. Get job runs.
* 11. Delete a job.
* 12. Delete a database.
* 13. Delete a crawler.
*/

```

```

public class GlueScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>\s

            Where:
                iam - The ARN of the IAM role that has AWS Glue and S3
permissions.\s
                s3Path - The Amazon Simple Storage Service (Amazon S3) target
that contains data (for example, CSV data).
                cron - A cron expression used to specify the schedule (i.e.,
cron(15 12 * * ? *)).
                dbName - The database name.\s
                crawlerName - The name of the crawler.\s
                jobName - The name you assign to this job definition.
                scriptLocation - The Amazon S3 path to a script that runs a job.
                locationUri - The location of the database
                bucketNameSc - The Amazon S3 bucket name used when creating a
job

            """;

        if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
        }

        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        String jobName = args[5];

```

```
String scriptLocation = args[6];
String locationUri = args[7];
String bucketNameSc = args[8];

Region region = Region.US_EAST_1;
GlueClient glueClient = GlueClient.builder()
    .region(region)
    .build();
System.out.println(DASHES);
System.out.println("Welcome to the AWS Glue scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create a database.");
createDatabase(glueClient, dbName, locationUri);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a crawler.");
createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get a crawler.");
getSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Start a crawler.");
startSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get a database.");
getSpecificDatabase(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("*** Wait 5 min for the tables to become available");
TimeUnit.MINUTES.sleep(5);
System.out.println("6. Get tables.");
String myTableName = getGlueTables(glueClient, dbName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("7. Create a job.");
createJob(glueClient, jobName, iam, scriptLocation);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Start a Job run.");
startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. List all jobs.");
getAllJobs(glueClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get job runs.");
getJobRuns(glueClient, jobName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Delete a job.");
deleteJob(glueClient, jobName);
System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
TimeUnit.MINUTES.sleep(5);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Delete a database.");
deleteDatabase(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Delete a crawler.");
deleteSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Successfully completed the AWS Glue Scenario");
System.out.println(DASHES);
}

public static void createDatabase(GlueClient glueClient, String dbName, String
locationUri) {
```

```
    try {
        DatabaseInput input = DatabaseInput.builder()
            .description("Built with the AWS SDK for Java V2")
            .name(dbName)
            .locationUri(locationUri)
            .build();

        CreateDatabaseRequest request = CreateDatabaseRequest.builder()
            .databaseInput(input)
            .build();

        glueClient.createDatabase(request);
        System.out.println(dbName + " was successfully created");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createGlueCrawler(GlueClient glueClient,
    String iam,
    String s3Path,
    String cron,
    String dbName,
    String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)
            .build();

        List<S3Target> targetList = new ArrayList<>();
        targetList.add(s3Target);
        CrawlerTargets targets = CrawlerTargets.builder()
            .s3Targets(targetList)
            .build();

        CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
            .databaseName(dbName)
            .name(crawlerName)
            .description("Created by the AWS Glue Java API")
            .targets(targets)
            .role(iam)
    }
```



```
        .schedule(cron)
        .build();

    glueClient.createCrawler(crawlerRequest);
    System.out.println(crawlerName + " was successfully created");

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
{
    try {
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        boolean ready = false;
        while (!ready) {
            GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
            String status = response.crawler().stateAsString();
            if (status.compareTo("READY") == 0) {
                ready = true;
            }
            Thread.sleep(3000);
        }

        System.out.println("The crawler is now ready");

    } catch (GlueException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
    try {
        StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
            .name(crawlerName)
            .build();
```

```
        glueClient.startCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully started!");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
    try {
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
            .name(databaseName)
            .build();

        GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
        Instant createDate = response.database().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the database is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String getGlueTables(GlueClient glueClient, String dbName) {
    String myTableName = "";
    try {
        GetTablesRequest tableRequest = GetTablesRequest.builder()
            .databaseName(dbName)
            .build();

        GetTablesResponse response = glueClient.getTables(tableRequest);
        List<Table> tables = response.tableList();
        if (tables.isEmpty()) {
```

```
        System.out.println("No tables were returned");
    } else {
        for (Table table : tables) {
            myTableName = table.name();
            System.out.println("Table name is: " + myTableName);
        }
    }

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return myTableName;
}

public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
    String outBucket) {
    try {
        Map<String, String> myMap = new HashMap<>();
        myMap.put("--input_database", inputDatabase);
        myMap.put("--input_table", inputTable);
        myMap.put("--output_bucket_url", outBucket);

        StartJobRunRequest runRequest = StartJobRunRequest.builder()
            .workerType(WorkerType.G_1_X)
            .numberOfWorkers(10)
            .arguments(myMap)
            .jobName(jobName)
            .build();

        StartJobRunResponse response = glueClient.startJobRun(runRequest);
        System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createJob(GlueClient glueClient, String jobName, String iam,
String scriptLocation) {
    try {
```

```
        JobCommand command = JobCommand.builder()
            .pythonVersion("3")
            .name("glueetl")
            .scriptLocation(scriptLocation)
            .build();

        CreateJobRequest jobRequest = CreateJobRequest.builder()
            .description("A Job created by using the AWS SDK for Java V2")
            .glueVersion("2.0")
            .workerType(WorkerType.G_1_X)
            .numberOfWorkers(10)
            .name(jobName)
            .role(iam)
            .command(command)
            .build();

        glueClient.createJob(jobRequest);
        System.out.println(jobName + " was successfully created.");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getAllJobs(GlueClient glueClient) {
    try {
        GetJobsRequest jobsRequest = GetJobsRequest.builder()
            .maxResults(10)
            .build();

        GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
        List<Job> jobs = jobsResponse.jobs();
        for (Job job : jobs) {
            System.out.println("Job name is : " + job.name());
            System.out.println("The job worker type is : " +
job.workerType().name());
        }

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
public static void getJobRuns(GlueClient glueClient, String jobName) {
    try {
        GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
            .jobName(jobName)
            .maxResults(20)
            .build();

        boolean jobDone = false;
        while (!jobDone) {
            GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
            List<JobRun> jobRuns = response.jobRuns();
            for (JobRun jobRun : jobRuns) {
                String jobState = jobRun.jobRunState().name();
                if (jobState.compareTo("SUCCEEDED") == 0) {
                    System.out.println(jobName + " has succeeded");
                    jobDone = true;

                } else if (jobState.compareTo("STOPPED") == 0) {
                    System.out.println("Job run has stopped");
                    jobDone = true;

                } else if (jobState.compareTo("FAILED") == 0) {
                    System.out.println("Job run has failed");
                    jobDone = true;

                } else if (jobState.compareTo("TIMEOUT") == 0) {
                    System.out.println("Job run has timed out");
                    jobDone = true;

                } else {
                    System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                    System.out.println("Job run Id is " + jobRun.id());
                    System.out.println("The Glue version is " +
jobRun.glueVersion());
                }
                TimeUnit.SECONDS.sleep(5);
            }
        }

    } catch (GlueException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
  
  public static void deleteJob(GlueClient glueClient, String jobName) {  
    try {  
      DeleteJobRequest jobRequest = DeleteJobRequest.builder()  
        .jobName(jobName)  
        .build();  
  
      glueClient.deleteJob(jobRequest);  
      System.out.println(jobName + " was successfully deleted");  
  
    } catch (GlueException e) {  
      System.err.println(e.awsErrorDetails().errorMessage());  
      System.exit(1);  
    }  
  }  
  
  public static void deleteDatabase(GlueClient glueClient, String databaseName) {  
    try {  
      DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()  
        .name(databaseName)  
        .build();  
  
      glueClient.deleteDatabase(request);  
      System.out.println(databaseName + " was successfully deleted");  
  
    } catch (GlueException e) {  
      System.err.println(e.awsErrorDetails().errorMessage());  
      System.exit(1);  
    }  
  }  
  
  public static void deleteSpecificCrawler(GlueClient glueClient, String  
crawlerName) {  
    try {  
      DeleteCrawlerRequest deleteCrawlerRequest =  
DeleteCrawlerRequest.builder()  
        .name(crawlerName)  
        .build();  
  
      glueClient.deleteCrawler(deleteCrawlerRequest);  
      System.out.println(crawlerName + " was deleted");  
    }  
  }  
}
```

```
        } catch (GlueException e) {  
            System.err.println(e.awsErrorDetails().errorMessage());  
            System.exit(1);  
        }  
    }  
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [CreateCrawler](#)
  - [CreateJob](#)
  - [DeleteCrawler](#)
  - [DeleteDatabase](#)
  - [DeleteJob](#)
  - [DeleteTable](#)
  - [GetCrawler](#)
  - [GetDatabase](#)
  - [GetDatabases](#)
  - [GetJob](#)
  - [GetJobRun](#)
  - [GetJobRuns](#)
  - [GetTables](#)
  - [ListJobs](#)
  - [StartCrawler](#)
  - [StartJobRun](#)

## HealthImaging contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with HealthImaging.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

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

## Tindakan

Penambahan sebuah tag ke sebuah sumber daya

Contoh kode berikut menunjukkan cara menambahkan tag ke HealthImaging sumber daya.

SDK for Java 2.x

```
public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
    String resourceArn,
    Map<String, String> tags) {
    try {
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tags(tags)
            .build();

        medicalImagingClient.tagResource(tagResourceRequest);

        System.out.println("Tags have been added to the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [TagResource](#) di Referensi AWS SDK for Java 2.x API.



**Note**

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

Salin satu set gambar

Contoh kode berikut menunjukkan cara menyalin set HealthImaging gambar.

SDK for Java 2.x

```
public static String copyMedicalImageSet(MedicalImagingClient
medicalImagingClient,
    String datastoreId,
    String imageSetId,
    String latestVersionId,
    String destinationImageSetId,
    String destinationVersionId) {

    try {
        CopySourceImageSetInformation copySourceImageSetInformation =
CopySourceImageSetInformation.builder()
            .latestVersionId(latestVersionId)
            .build();

        CopyImageSetInformation.Builder copyImageSetBuilder =
CopyImageSetInformation.builder()
            .sourceImageSet(copySourceImageSetInformation);

        if (destinationImageSetId != null) {
            copyImageSetBuilder =
copyImageSetBuilder.destinationImageSet(CopyDestinationImageSet.builder()
                .imageSetId(destinationImageSetId)
                .latestVersionId(destinationVersionId)
                .build());
        }

        CopyImageSetRequest copyImageSetRequest = CopyImageSetRequest.builder()
            .datastoreId(datastoreId)
            .sourceImageSetId(imageSetId)
            .copyImageSetInformation(copyImageSetBuilder.build())
            .build();
```

```

        CopyImageSetResponse response =
medicalImagingClient.copyImageSet(copyImageSetRequest);

        return response.destinationImageSetProperties().imageSetId();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

```

- Untuk detail API, lihat [CopyImageSet](#) di Referensi AWS SDK for Java 2.x API.

#### Note

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

## Buat penyimpanan data

Contoh kode berikut menunjukkan cara membuat penyimpanan HealthImaging data.

### SDK for Java 2.x

```

public static String createMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
        String datastoreName) {
    try {
        CreateDatastoreRequest datastoreRequest =
CreateDatastoreRequest.builder()
                .datastoreName(datastoreName)
                .build();

        CreateDatastoreResponse response =
medicalImagingClient.createDatastore(datastoreRequest);
        return response.datastoreId();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

```
        return "";  
    }
```

- Untuk detail API, lihat [CreateDatastore](#) di Referensi AWS SDK for Java 2.x API.

#### Note

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

## Hapus penyimpanan data

Contoh kode berikut menunjukkan cara menghapus penyimpanan HealthImaging data.

### SDK for Java 2.x

```
public static void deleteMedicalImagingDatastore(MedicalImagingClient  
medicalImagingClient,  
        String datastoreID) {  
    try {  
        DeleteDatastoreRequest datastoreRequest =  
DeleteDatastoreRequest.builder()  
            .datastoreId(datastoreID)  
            .build();  
        medicalImagingClient.deleteDatastore(datastoreRequest);  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Untuk detail API, lihat [DeleteDatastore](#) di Referensi AWS SDK for Java 2.x API.

#### Note

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

## Hapus satu set gambar

Contoh kode berikut menunjukkan cara menghapus set HealthImaging gambar.

### SDK for Java 2.x

```
public static void deleteMedicalImageSet(MedicalImagingClient
medicalImagingClient,
    String datastoreId,
    String imagesetId) {
    try {
        DeleteImageSetRequest deleteImageSetRequest =
DeleteImageSetRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId)
            .build();

        medicalImagingClient.deleteImageSet(deleteImageSetRequest);

        System.out.println("The image set was deleted.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DeleteImageSet](#) di Referensi AWS SDK for Java 2.x API.

#### Note

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

## Dapatkan bingkai gambar

Contoh kode berikut menunjukkan cara mendapatkan bingkai gambar.

## SDK for Java 2.x

```
public static void getMedicalImageSetFrame(MedicalImagingClient
medicalImagingClient,
    String destinationPath,
    String datastoreId,
    String imagesetId,
    String imageFrameId) {

    try {
        GetImageFrameRequest getImageSetMetadataRequest =
        GetImageFrameRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId)

            .imageFrameInformation(ImageFrameInformation.builder()
                .imageFrameId(imageFrameId)
                .build())
            .build();

        medicalImagingClient.getImageFrame(getImageSetMetadataRequest,
        FileSystems.getDefault().getPath(destinationPath));

        System.out.println("Image frame downloaded to " +
        destinationPath);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [GetImageFrame](#) di Referensi AWS SDK for Java 2.x API.

**Note**

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

## Dapatkan properti penyimpanan data

Contoh kode berikut menunjukkan cara mendapatkan properti penyimpanan HealthImaging data.

### SDK for Java 2.x

```
public static DatastoreProperties getMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
    String datastoreID) {
    try {
        GetDatastoreRequest datastoreRequest = GetDatastoreRequest.builder()
            .datastoreId(datastoreID)
            .build();
        GetDatastoreResponse response =
medicalImagingClient.getDatastore(datastoreRequest);
        return response.datastoreProperties();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Untuk detail API, lihat [GetDatastore](#) di Referensi AWS SDK for Java 2.x API.

#### Note

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

## Dapatkan properti set gambar

Contoh kode berikut menunjukkan bagaimana untuk mendapatkan properti HealthImaging image set.

### SDK for Java 2.x

```
public static GetImageSetResponse getMedicalImageSet(MedicalImagingClient
medicalImagingClient,
    String datastoreId,
```

```
        String imagesetId,
        String versionId) {
    try {
        GetImageSetRequest.Builder getImageSetRequestBuilder =
        GetImageSetRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId);

        if (versionId != null) {
            getImageSetRequestBuilder =
            getImageSetRequestBuilder.versionId(versionId);
        }

        return
        medicalImagingClient.getImageSet(getImageSetRequestBuilder.build());
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Untuk detail API, lihat [GetImageSet](#) di Referensi AWS SDK for Java 2.x API.

#### Note

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

Dapatkan properti pekerjaan impor

Contoh kode berikut menunjukkan cara mendapatkan properti pekerjaan impor.

SDK for Java 2.x

```
public static DICOMImportJobProperties getDicomImportJob(MedicalImagingClient
medicalImagingClient,
    String datastoreId,
    String jobId) {
```

```
    try {
        GetDicomImportJobRequest getDicomImportJobRequest =
        GetDicomImportJobRequest.builder()
            .datastoreId(datastoreId)
            .jobId(jobId)
            .build();

        GetDicomImportJobResponse response =
        medicalImagingClient.getDICOMImportJob(getDicomImportJobRequest);
        return response.jobProperties();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Untuk detail API, lihat [GetDicom ImportJob](#) di Referensi AWS SDK for Java 2.x API.

#### Note

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

Dapatkan metadata untuk kumpulan gambar

Contoh kode berikut menunjukkan cara mendapatkan metadata untuk kumpulan HealthImaging gambar.

SDK for Java 2.x

```
public static void getMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
    String destinationPath,
    String datastoreId,
    String imagesetId,
    String versionId) {

    try {
```



```

        GetImageSetMetadataRequest.Builder getImageSetMetadataRequestBuilder =
        GetImageSetMetadataRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId);

        if (versionId != null) {
            getImageSetMetadataRequestBuilder =
        getImageSetMetadataRequestBuilder.versionId(versionId);
        }

        medicalImagingClient.getImageSetMetadata(getImageSetMetadataRequestBuilder.build(),
            FileSystems.getDefault().getPath(destinationPath));

        System.out.println("Metadata downloaded to " + destinationPath);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Untuk detail API, lihat [GetImageSetMetadata](#) di Referensi AWS SDK for Java 2.x API.

#### Note

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

## Impor data massal ke penyimpanan data

Contoh kode berikut menunjukkan cara mengimpor data massal ke penyimpanan HealthImaging data.

### SDK for Java 2.x

```

    public static String startDicomImportJob(MedicalImagingClient
        medicalImagingClient,
            String jobName,
            String datastoreId,
            String dataAccessRoleArn,

```

```

        String inputS3Uri,
        String outputS3Uri) {

    try {
        StartDicomImportJobRequest startDicomImportJobRequest =
StartDicomImportJobRequest.builder()
        .jobName(jobName)
        .datastoreId(datastoreId)
        .dataAccessRoleArn(dataAccessRoleArn)
        .inputS3Uri(inputS3Uri)
        .outputS3Uri(outputS3Uri)
        .build();

        StartDicomImportJobResponse response =
medicalImagingClient.startDICOMImportJob(startDicomImportJobRequest);
        return response.jobId();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

```

- Untuk detail API, lihat [StartDicom ImportJob](#) di AWS SDK for Java 2.x Referensi API.

#### Note

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

## Daftar penyimpanan data

Contoh kode berikut menunjukkan cara daftar penyimpanan HealthImaging data.

### SDK for Java 2.x

```

public static List<DatastoreSummary>
listMedicalImagingDatastores(MedicalImagingClient medicalImagingClient) {
    try {
        ListDatastoresRequest datastoreRequest = ListDatastoresRequest.builder()
        .build();
    }
}

```

```

        ListDatastoresIterable responses =
medicalImagingClient.listDatastoresPaginator(datastoreRequest);
        List<DatastoreSummary> datastoreSummaries = new ArrayList<>();

        responses.stream().forEach(response ->
datastoreSummaries.addAll(response.datastoreSummaries()));

        return datastoreSummaries;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}

```

- Untuk detail API, lihat [ListDatastores](#) di Referensi AWS SDK for Java 2.x API.

#### Note

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

## Daftar versi set gambar

Contoh kode berikut menunjukkan cara daftar versi set HealthImaging gambar.

### SDK for Java 2.x

```

    public static List<ImageSetProperties>
listMedicalImageSetVersions(MedicalImagingClient medicalImagingClient,
        String datastoreId,
        String imagesetId) {
    try {
        ListImageSetVersionsRequest getImageSetRequest =
ListImageSetVersionsRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId)
            .build();

        ListImageSetVersionsIterable responses = medicalImagingClient

```

```

        .listImageSetVersionsPaginator(getImageSetRequest);
        List<ImageSetProperties> imageSetProperties = new ArrayList<>();
        responses.stream().forEach(response ->
imageSetProperties.addAll(response.imageSetPropertiesList()));

        return imageSetProperties;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}

```

- Untuk detail API, lihat [ListImageSetVersions](#) di Referensi AWS SDK for Java 2.x API.

#### Note

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

## Daftar pekerjaan impor untuk penyimpanan data

Contoh kode berikut menunjukkan cara daftar pekerjaan impor untuk penyimpanan HealthImaging data.

### SDK for Java 2.x

```

public static List<DICOMImportJobSummary>
listDicomImportJobs(MedicalImagingClient medicalImagingClient,
String datastoreId) {

    try {
        ListDicomImportJobsRequest listDicomImportJobsRequest =
ListDicomImportJobsRequest.builder()
            .datastoreId(datastoreId)
            .build();

        ListDicomImportJobsResponse response =
medicalImagingClient.listDICOMImportJobs(listDicomImportJobsRequest);
        return response.jobSummaries();
    }
}

```

```
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return new ArrayList<>();
}
```

- Untuk detail API, lihat [ListDicom ImportJobs](#) di Referensi AWS SDK for Java 2.x API.

### Note

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

## Membuat daftar tanda untuk sumber daya

Contoh kode berikut menunjukkan bagaimana untuk daftar tag untuk HealthImaging sumber daya.

### SDK for Java 2.x

```
public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
    String resourceArn) {
    try {
        ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
            .resourceArn(resourceArn)
            .build();

        return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Untuk detail API, lihat [ListTagsForResource](#) di Referensi AWS SDK for Java 2.x API.

 Note

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

## Menghapus tag dari sumber daya

Contoh kode berikut menunjukkan cara menghapus tag dari HealthImaging sumber daya.


### SDK for Java 2.x

```
public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
        String resourceArn,
        Collection<String> tagKeys) {
    try {
        UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tagKeys(tagKeys)
            .build();

        medicalImagingClient.untagResource(untagResourceRequest);

        System.out.println("Tags have been removed from the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [UntagResource](#) di Referensi AWS SDK for Java 2.x API.

 Note

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

## Cari set gambar

Contoh kode berikut menunjukkan cara mencari set HealthImaging gambar.

### SDK for Java 2.x

Fungsi utilitas untuk mencari set gambar.

```
public static List<ImageSetsMetadataSummary> searchMedicalImagingImageSets(
    MedicalImagingClient medicalImagingClient,
    String datastoreId, List<SearchFilter> searchFilters) {
    try {
        SearchImageSetsRequest datastoreRequest =
SearchImageSetsRequest.builder()
                        .datastoreId(datastoreId)

        .searchCriteria(SearchCriteria.builder().filters(searchFilters).build())
                        .build();
        SearchImageSetsIterable responses = medicalImagingClient
                        .searchImageSetsPaginator(datastoreRequest);
        List<ImageSetsMetadataSummary> imageSetsMetadataSummaries =
new ArrayList<>();

        responses.stream().forEach(response ->
imageSetsMetadataSummaries

        .addAll(response.imageSetsMetadataSummaries()));

        return imageSetsMetadataSummaries;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

Kasus penggunaan #1: operator EQUAL.

```
List<SearchFilter> searchFilters =
Collections.singletonList(SearchFilter.builder()
                        .operator(Operator.EQUAL)
```

```

        .values(SearchByAttributeValue.builder()
                .dicomPatientId(patientId)
                .build())
        .build());

    List<ImageSetsMetadataSummary> imageSetsMetadataSummaries =
searchMedicalImagingImageSets(
        medicalImagingClient,
        datastoreId, searchFilters);
    if (imageSetsMetadataSummaries != null) {
        System.out.println("The image sets for patient " + patientId
+ " are:\n"
                + imageSetsMetadataSummaries);
        System.out.println();
    }

```

Kasus penggunaan #2: ANTARA operator menggunakan DICOM StudyDate dan StudyTime DICOM.

```

        DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("yyyyMMdd");
        searchFilters = Collections.singletonList(SearchFilter.builder()
                .operator(Operator.BETWEEN)
                .values(SearchByAttributeValue.builder()

.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()

.dicomStudyDate("19990101")

.dicomStudyTime("000000.000")

                .build())
                .build(),
                SearchByAttributeValue.builder()

.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()

.dicomStudyDate((LocalDate.now()

                .format(formatter)))

.dicomStudyTime("000000.000")

```



```

.build()

                                .build()

                                .build());

    imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                                datastoreId, searchFilters);
    if (imageSetsMetadataSummaries != null) {
        System.out.println(
            "The image sets searched with BETWEEN
operator using DICOMStudyDate and DICOMStudyTime are:\n"
            +
            imageSetsMetadataSummaries);
        System.out.println();
    }

```

Kasus penggunaan #3: ANTARA operator menggunakan createDat. Studi waktu sebelumnya bertahan.

```

        searchFilters = Collections.singletonList(SearchFilter.builder()
            .operator(Operator.BETWEEN)
            .values(SearchByAttributeValue.builder()

                .createdAt(Instant.parse("1985-04-12T23:20:50.52Z"))
                    .build(),
                SearchByAttributeValue.builder()

                .createdAt(Instant.now())
                    .build())
            .build());

    imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                                datastoreId, searchFilters);
    if (imageSetsMetadataSummaries != null) {
        System.out.println("The image sets searched with BETWEEN
operator using createdAt are:\n "
            + imageSetsMetadataSummaries);
        System.out.println();
    }

```

- Untuk detail API, lihat [SearchImageSets](#) di Referensi AWS SDK for Java 2.x API.

 Note

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

## Perbarui metadata set gambar

Contoh kode berikut menunjukkan cara memperbarui HealthImaging metadata set gambar.

### SDK for Java 2.x

```
public static void updateMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
        String datastoreId,
        String imagesetId,
        String versionId,
        MetadataUpdates metadataUpdates) {
    try {
        UpdateImageSetMetadataRequest updateImageSetMetadataRequest
= UpdateImageSetMetadataRequest
                .builder()
                .datastoreId(datastoreId)
                .imageSetId(imagesetId)
                .latestVersionId(versionId)
                .updateImageSetMetadataUpdates(metadataUpdates)
                .build();

        medicalImagingClient.updateImageSetMetadata(updateImageSetMetadataRequest);

        System.out.println("The image set metadata was updated");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [UpdateImageSetMetadadi Referensi AWS SDK for Java 2.x API](#).

 Note

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

## Skenario

### Menandai penyimpanan data

Contoh kode berikut menunjukkan cara menandai penyimpanan HealthImaging data.

### SDK for Java 2.x

Untuk menandai penyimpanan data.

```
final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

TagResource.tagMedicalImagingResource(medicalImagingClient,
datastoreArn,
    ImmutableMap.of("Deployment", "Development"));
```

Fungsi utilitas untuk menandai sumber daya.

```
public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
    String resourceArn,
    Map<String, String> tags) {
    try {
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tags(tags)
            .build();

        medicalImagingClient.tagResource(tagResourceRequest);

        System.out.println("Tags have been added to the resource.");
    } catch (MedicalImagingException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Untuk daftar tag untuk penyimpanan data.

```
        final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

        ListTagsForResourceResponse result =
ListTagsForResource.listMedicalImagingResourceTags(
            medicalImagingClient,
            datastoreArn);
        if (result != null) {
            System.out.println("Tags for resource: " + result.tags());
        }
```

Fungsi utilitas untuk daftar tag sumber daya.

```
    public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
    String resourceArn) {
        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
                .resourceArn(resourceArn)
                .build();

            return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

Untuk menghapus tag penyimpanan data.

```

        final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

        UntagResource.untagMedicalImagingResource(medicalImagingClient,
datastoreArn,
                Collections.singletonList("Deployment"));

```

Fungsi utilitas untuk membuka tag sumber daya.

```

public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
        String resourceArn,
        Collection<String> tagKeys) {
    try {
        UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
                .resourceArn(resourceArn)
                .tagKeys(tagKeys)
                .build();

        medicalImagingClient.untagResource(untagResourceRequest);

        System.out.println("Tags have been removed from the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [ListTagsForResource](#)
  - [TagResource](#)
  - [UntagResource](#)

#### Note

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

## Menandai set gambar

Contoh kode berikut menunjukkan cara menandai set HealthImaging gambar.

### SDK for Java 2.x

Untuk menandai set gambar.

```
final String imageSetArn = "arn:aws:medical-imaging:us-  
east-1:123456789012:datastore/12345678901234567890123456789012/  
imageset/12345678901234567890123456789012";  
  
TagResource.tagMedicalImagingResource(medicalImagingClient,  
imageSetArn,  
ImmutableMap.of("Deployment", "Development"));
```

Fungsi utilitas untuk menandai sumber daya.

```
public static void tagMedicalImagingResource(MedicalImagingClient  
medicalImagingClient,  
String resourceArn,  
Map<String, String> tags) {  
    try {  
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()  
            .resourceArn(resourceArn)  
            .tags(tags)  
            .build();  
  
        medicalImagingClient.tagResource(tagResourceRequest);  
  
        System.out.println("Tags have been added to the resource.");  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

Untuk mencantumkan tag untuk kumpulan gambar.

```
        final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

        ListTagsForResourceResponse result =
ListTagsForResource.listMedicalImagingResourceTags(
            medicalImagingClient,
            imageSetArn);
        if (result != null) {
            System.out.println("Tags for resource: " + result.tags());
        }
    }
```

Fungsi utilitas untuk daftar tag sumber daya.

```
    public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
    String resourceArn) {
        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
                .resourceArn(resourceArn)
                .build();

            return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
}
```

Untuk menghapus tag set gambar.


```
        final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
```

```
UntagResource.untagMedicalImagingResource(medicalImagingClient,  
imageSetArn,  
Collections.singletonList("Deployment"));
```

Fungsi utilitas untuk membuka tag sumber daya.

```
public static void untagMedicalImagingResource(MedicalImagingClient  
medicalImagingClient,  
String resourceArn,  
Collection<String> tagKeys) {  
    try {  
        UntagResourceRequest untagResourceRequest =  
UntagResourceRequest.builder()  
            .resourceArn(resourceArn)  
            .tagKeys(tagKeys)  
            .build();  
  
        medicalImagingClient.untagResource(untagResourceRequest);  
  
        System.out.println("Tags have been removed from the resource.");  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [ListTagsForResource](#)
  - [TagResource](#)
  - [UntagResource](#)

 Note

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



## Contoh IAM menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with IAM.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Memulai

Halo IAM

Contoh kode berikut menunjukkan bagaimana memulai menggunakan IAM.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.ListPoliciesResponse;
import software.amazon.awssdk.services.iam.model.Policy;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class HelloIAM {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listPolicies(iam);
    }

    public static void listPolicies(IamClient iam) {
        ListPoliciesResponse response = iam.listPolicies();
        List<Policy> polList = response.policies();
        polList.forEach(policy -> {
            System.out.println("Policy Name: " + policy.policyName());
        });
    }
}
```

- Untuk detail API, lihat [ListPolicies](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

Lampirkan kebijakan ke peran

Contoh kode berikut menunjukkan cara melampirkan kebijakan IAM ke peran.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AttachRolePolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <roleName> <policyArn>\s

            Where:
                roleName - A role name that you can obtain from the AWS
Management Console.\s
                policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();
    }
}
```

```
        attachIAMRolePolicy(iam, roleName, policyArn);
        iam.close();
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                .roleName(roleName)
                .build();

            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

            // Ensure that the policy is not attached to this role
            String polArn = "";
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
this role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policyArn)
                .build();

            iam.attachRolePolicy(attachRequest);

            System.out.println("Successfully attached policy " + policyArn +
                " to role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

```
}  
}
```

- Untuk detail API, lihat [AttachRolePolicy](#) di Referensi AWS SDK for Java 2.x API.

## Buat kebijakan

Contoh kode berikut menunjukkan cara membuat kebijakan IAM.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.core.waiters.WaiterResponse;  
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;  
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;  
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;  
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;  
import software.amazon.awssdk.services.iam.model.IamException;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.iam.IamClient;  
import software.amazon.awssdk.services.iam.waiters.IamWaiter;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class CreatePolicy {  
  
    public static final String PolicyDocument = "{" +  
        "  \"Version\": \"2012-10-17\", \" +  
        "  \"Statement\": [\" +  
        "    {\" +
```

```

    "        \"Effect\": \"Allow\", \" +
    "        \"Action\": [\" +
    "            \"dynamodb:DeleteItem\", \" +
    "            \"dynamodb:GetItem\", \" +
    "            \"dynamodb:PutItem\", \" +
    "            \"dynamodb:Scan\", \" +
    "            \"dynamodb:UpdateItem\"\" +
    "        ], \" +
    "        \"Resource\": \"*\")\" +
    "    }\" +
    "]" +
    "};

```

```

public static void main(String[] args) {

    final String usage = ""
        Usage:
            CreatePolicy <policyName>\s

        Where:
            policyName - A unique policy name.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String policyName = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    String result = createIAMPolicy(iam, policyName);
    System.out.println("Successfully created a policy with this ARN value: " +
result);
    iam.close();
}

public static String createIAMPolicy(IamClient iam, String policyName) {
    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();

```

```
        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument)
            .build();

        CreatePolicyResponse response = iam.createPolicy(request);

        // Wait until the policy is created.
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Untuk detail API, lihat [CreatePolicy](#) di Referensi AWS SDK for Java 2.x API.

## Membuat peran

Contoh kode berikut menunjukkan cara membuat peran IAM.

SDK for Java 2.x

### Note

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

```
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import software.amazon.awssdk.services.iam.model.CreateRoleRequest;
import software.amazon.awssdk.services.iam.model.CreateRoleResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import java.io.FileReader;

/*
 * This example requires a trust policy document. For more information, see:
 * https://aws.amazon.com/blogs/security/how-to-use-trust-policies-with-iam-roles/
 *
 * In addition, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateRole {
    public static void main(String[] args) throws Exception {
        final String usage = ""
            Usage:
                <rolename> <fileLocation>\s

                Where:
                    rolename - The name of the role to create.\s
                    fileLocation - The location of the JSON document that represents
the trust policy.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String rolename = args[0];
        String fileLocation = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
```



```

        .build();

        String result = createIAMRole(iam, rolename, fileLocation);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMRole(IamClient iam, String rolename, String
fileLocation) throws Exception {
        try {
            JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
            CreateRoleRequest request = CreateRoleRequest.builder()
                .roleName(rolename)
                .assumeRolePolicyDocument(jsonObject.toJSONString())
                .description("Created using the AWS SDK for Java")
                .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " + response.role().arn());

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static Object readJsonSimpleDemo(String filename) throws Exception {
        FileReader reader = new FileReader(filename);
        JSONParser jsonParser = new JSONParser();
        return jsonParser.parse(reader);
    }
}

```

- Untuk detail API, lihat [CreateRole](#) di Referensi AWS SDK for Java 2.x API.

## Buat pengguna

Contoh kode berikut menunjukkan cara membuat pengguna IAM.

**⚠ 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](#).

## SDK for Java 2.x

**ℹ Note**

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

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUser {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <username>\s

                Where:
```

```
        username - The name of the user to create.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String username = args[0];
    Region region = Region.AWS_GLOBAL;
    IAMClient iam = IAMClient.builder()
        .region(region)
        .build();

    String result = createIAMUser(iam, username);
    System.out.println("Successfully created user: " + result);
    iam.close();
}

public static String createIAMUser(IAMClient iam, String username) {
    try {
        // Create an IAMWaiter object.
        IAMWaiter iamWaiter = iam.waiter();

        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        CreateUserResponse response = iam.createUser(request);

        // Wait until the user is created.
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user().userName();

    } catch (IAMException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        return "";  
    }  
}
```

- Untuk detail API, lihat [CreateUser](#) di Referensi AWS SDK for Java 2.x API.

## Buat kunci akses

Contoh kode berikut menunjukkan cara membuat kunci akses IAM.

### 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](#).

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;  
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.iam.IamClient;  
import software.amazon.awssdk.services.iam.model.IamException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class CreateAccessKey {
```

```
public static void main(String[] args) {
    final String usage = ""

        Usage:
        <user>\s

        Where:
        user - An AWS IAM user that you can obtain from the AWS
Management Console.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String user = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    String keyId = createIAMAccessKey(iam, user);
    System.out.println("The Key Id is " + keyId);
    iam.close();
}

public static String createIAMAccessKey(IamClient iam, String user) {
    try {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(user)
            .build();

        CreateAccessKeyResponse response = iam.createAccessKey(request);
        return response.accessKey().accessKeyId();


    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Untuk detail API, lihat [CreateAccessKey](#) di Referensi AWS SDK for Java 2.x API.

Buat alias untuk akun

Contoh kode berikut menunjukkan cara membuat alias untuk akun IAM.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.services.iam.model.CreateAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAccountAlias {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <alias>\s

                Where:
                    alias - The account alias to create (for example, myawsaccount).
\s

            """;

        if (args.length != 1) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String alias = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    createIAMAccountAlias(iam, alias);
    iam.close();
    System.out.println("Done");
}

public static void createIAMAccountAlias(IamClient iam, String alias) {
    try {
        CreateAccountAliasRequest request = CreateAccountAliasRequest.builder()
            .accountAlias(alias)
            .build();

        iam.createAccountAlias(request);
        System.out.println("Successfully created account alias: " + alias);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [CreateAccountAlias](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus kebijakan

Contoh kode berikut menunjukkan cara menghapus kebijakan IAM.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.services.iam.model.DeletePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeletePolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <policyARN>\s

            Where:
                policyARN - A policy ARN value to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyARN = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();
```



```
        deleteIAMPolicy(iam, policyARN);
        iam.close();
    }

    public static void deleteIAMPolicy(IamClient iam, String policyARN) {
        try {
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                .policyArn(policyARN)
                .build();

            iam.deletePolicy(request);
            System.out.println("Successfully deleted the policy");

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Untuk detail API, lihat [DeletePolicy](#) di Referensi AWS SDK for Java 2.x API.

## Hapus pengguna

Contoh kode berikut menunjukkan cara menghapus pengguna IAM.

### 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](#).

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteUser {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <userName>\s

                Where:
                userName - The name of the user to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();
```

```
        deleteIAMUser(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                .userName(userName)
                .build();

            iam.deleteUser(request);
            System.out.println("Successfully deleted IAM user " + userName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DeleteUser](#) di Referensi AWS SDK for Java 2.x API.

## Hapus kunci akses

Contoh kode berikut menunjukkan cara menghapus kunci akses IAM.

### 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](#).

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAccessKey {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <username> <accessKey>\s

            Where:
                username - The name of the user.\s
                accessKey - The access key ID for the secret access key you want
to delete.\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessKey = args[1];
        Region region = Region.AWS_GLOBAL;
```

```
IamClient iam = IamClient.builder()
    .region(region)
    .build();
deleteKey(iam, username, accessKey);
iam.close();
}

public static void deleteKey(IamClient iam, String username, String accessKey) {
    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DeleteAccessKey](#) di Referensi AWS SDK for Java 2.x API.

## Hapus alias akun

Contoh kode berikut menunjukkan cara menghapus alias akun IAM.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.services.iam.model.DeleteAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAccountAlias {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <alias>\s

                Where:
                alias - The account alias to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        deleteIAMAccountAlias(iam, alias);
        iam.close();
    }

    public static void deleteIAMAccountAlias(IamClient iam, String alias) {
        try {
            DeleteAccountAliasRequest request = DeleteAccountAliasRequest.builder()
                .accountAlias(alias)
                .build();

            iam.deleteAccountAlias(request);
        }
    }
}
```

```
        System.out.println("Successfully deleted account alias " + alias);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
}
```

- Untuk detail API, lihat [DeleteAccountAlias](#) di Referensi AWS SDK for Java 2.x API.

Lepaskan kebijakan dari peran

Contoh kode berikut menunjukkan cara melepaskan kebijakan IAM dari peran.

SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetachRolePolicy {
    public static void main(String[] args) {
        final String usage = ""
```

```

        Usage:
            <roleName> <policyArn>\s

        Where:
            roleName - A role name that you can obtain from the AWS
Management Console.\s
            policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
            """";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();
        detachPolicy(iam, roleName, policyArn);
        System.out.println("Done");
        iam.close();
    }

    public static void detachPolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policyArn)
                .build();

            iam.detachRolePolicy(request);
            System.out.println("Successfully detached policy " + policyArn +
                " from role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```



- Untuk detail API, lihat [DetachRolePolicy](#) di Referensi AWS SDK for Java 2.x API.

## Daftar kunci akses pengguna

Contoh kode berikut menunjukkan bagaimana untuk daftar kunci akses IAM pengguna.

### 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](#).

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListAccessKeys {
    public static void main(String[] args) {
        final String usage = ""
```

```
Usage:
    <userName>\s

Where:
    userName - The name of the user for which access keys are
retrieved.\s
    """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String userName = args[0];
Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .build();

listKeys(iam, userName);
System.out.println("Done");
iam.close();
}

public static void listKeys(IamClient iam, String userName) {
    try {
        boolean done = false;
        String newMarker = null;

        while (!done) {
            ListAccessKeysResponse response;

            if (newMarker == null) {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .build();

                response = iam.listAccessKeys(request);
            } else {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .marker(newMarker)
```

```
        .build();

        response = iam.listAccessKeys(request);
    }

    for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
        System.out.format("Retrieved access key %s",
metadata.accessKeyId());
    }

    if (!response.isTruncated()) {
        done = true;
    } else {
        newMarker = response.marker();
    }
}

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
}
```

- Untuk detail API, lihat [ListAccessKeys](#) di Referensi AWS SDK for Java 2.x API.

## Daftar alias akun

Contoh kode berikut menunjukkan bagaimana untuk daftar alias akun IAM.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccountAliasesResponse;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListAccountAliases {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listAliases(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAliases(IamClient iam) {
        try {
            ListAccountAliasesResponse response = iam.listAccountAliases();
            for (String alias : response.accountAliases()) {
                System.out.printf("Retrieved account alias %s", alias);
            }
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListAccountAliases](#) di Referensi AWS SDK for Java 2.x API.

## Daftar pengguna

Contoh kode berikut menunjukkan bagaimana untuk daftar pengguna IAM.

**⚠ 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](#).

## SDK for Java 2.x

**ℹ Note**

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

```
import software.amazon.awssdk.services.iam.model.AttachedPermissionsBoundary;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.User;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUsers {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listAllUsers(iam);
        System.out.println("Done");
        iam.close();
    }
}
```

```
}

public static void listAllUsers(IamClient iam) {
    try {
        boolean done = false;
        String newMarker = null;
        while (!done) {
            ListUsersResponse response;
            if (newMarker == null) {
                ListUsersRequest request = ListUsersRequest.builder().build();
                response = iam.listUsers(request);
            } else {
                ListUsersRequest request = ListUsersRequest.builder()
                    .marker(newMarker)
                    .build();

                response = iam.listUsers(request);
            }

            for (User user : response.users()) {
                System.out.format("\n Retrieved user %s", user.userName());
                AttachedPermissionsBoundary permissionsBoundary =
user.permissionsBoundary();
                if (permissionsBoundary != null)
                    System.out.format("\n Permissions boundary details %s",
permissionsBoundary.permissionsBoundaryTypeAsString());
            }

            if (!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [ListUsers](#) di Referensi AWS SDK for Java 2.x API.

## Memperbarui pengguna

Contoh kode berikut menunjukkan cara memperbarui pengguna IAM.

### 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](#).

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateUser {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
            <curName> <newName>\s
```

```
        Where:
            curName - The current user name.\s
            newName - An updated user name.\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String curName = args[0];
    String newName = args[1];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    updateIAMUser(iam, curName, newName);
    System.out.println("Done");
    iam.close();
}

public static void updateIAMUser(IamClient iam, String curName, String newName)
{
    try {
        UpdateUserRequest request = UpdateUserRequest.builder()
            .userName(curName)
            .newUserName(newName)
            .build();

        iam.updateUser(request);
        System.out.printf("Successfully updated user to username %s", newName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [UpdateUser](#) di Referensi AWS SDK for Java 2.x API.



## Perbarui kunci akses

Contoh kode berikut menunjukkan cara memperbarui kunci akses IAM.

### 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](#).

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateAccessKey {

    private static StatusType statusType;

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <username> <accessId> <status>\s
```

```

        Where:
            username - The name of the user whose key you want to update.\s
            accessId - The access key ID of the secret access key you want
to update.\s
            status - The status you want to assign to the secret access key.
\s
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String username = args[0];
    String accessId = args[1];
    String status = args[2];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    updateKey(iam, username, accessId, status);
    System.out.println("Done");
    iam.close();
}

public static void updateKey(IamClient iam, String username, String accessId,
String status) {
    try {
        if (status.toLowerCase().equalsIgnoreCase("active")) {
            statusType = StatusType.ACTIVE;
        } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
            statusType = StatusType.INACTIVE;
        } else {
            statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
        }

        UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
            .accessKeyId(accessId)
            .userName(username)
            .status(statusType)
            .build();

```

```
        iam.updateAccessKey(request);
        System.out.printf("Successfully updated the status of access key %s to"
+
            "status %s for user %s", accessId, status, username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [UpdateAccessKey](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

### Membangun dan mengelola layanan yang tangguh

Contoh kode berikut menunjukkan cara membuat layanan web load-balanced yang mengembalikan rekomendasi buku, film, dan lagu. Contoh ini menunjukkan cara layanan tersebut merespons kegagalan, serta cara merestrukturisasi layanan agar lebih tangguh ketika terjadi kegagalan.

- Menggunakan grup Amazon EC2 Auto Scaling untuk membuat instans Amazon Elastic Compute Cloud (Amazon EC2) berdasarkan templat peluncuran dan menyimpan sejumlah instans dalam rentang yang ditentukan.
- Menangani dan mendistribusikan permintaan HTTP dengan Elastic Load Balancing.
- Memantau kondisi instans dalam grup Auto Scaling dan meneruskan permintaan hanya ke instans yang sehat.
- Menjalankan server web Python pada setiap instans EC2 untuk menangani permintaan HTTP. Server web merespons dengan memberikan rekomendasi dan melakukan pemeriksaan kondisi.
- Menyimulasikan layanan yang direkomendasikan dengan tabel Amazon DynamoDB.
- Kontrol respons server web terhadap permintaan dan pemeriksaan kesehatan dengan memperbarui AWS Systems Manager parameter.

## SDK for Java 2.x

 Note

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

Menjalankan skenario interaktif di prompt perintah.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\\
\\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\\
\\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\\0", "-");
```

```
public static void main(String[] args) throws IOException, InterruptedException
{
    Scanner in = new Scanner(System.in);
    Database database = new Database();
    AutoScaler autoScaler = new AutoScaler();
    LoadBalancer loadBalancer = new LoadBalancer();

    System.out.println(DASHES);
    System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("A - SETUP THE RESOURCES");
    System.out.println("Press Enter when you're ready to start deploying
resources.");
    in.nextLine();
    deploy(loadBalancer);
    System.out.println(DASHES);
    System.out.println(DASHES);
    System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    demo(loadBalancer);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("C - DELETE THE RESOURCES");
    System.out.println("""
        This concludes the demo of how to build and manage a resilient
service.
        To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
        that were created for this demo.
        """);

    System.out.println("\n Do you want to delete the resources (y/n)? ");
    String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

    if (userInput.equals("y")) {
        // Delete resources here
        deleteResources(loadBalancer, autoScaler, database);
        System.out.println("Resources deleted.");
    } else {
```

```

        System.out.println("""
            Okay, we'll leave the resources intact.
            Don't forget to delete them when you're done with them or you
might incur unexpected charges.
            """);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The example has completed. ");
    System.out.println("\n Thanks for watching!");
    System.out.println(DASHES);
}

// Deletes the AWS resources used in this example.
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
    throws IOException, InterruptedException {
    loadBalancer.deleteLoadBalancer(lbName);
    System.out.println("*** Wait 30 secs for resource to be deleted");
    TimeUnit.SECONDS.sleep(30);
    loadBalancer.deleteTargetGroup(targetGroupName);
    autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
    autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
    autoScaler.deleteTemplate(templateName);
    database.deleteTable(tableName);
}

private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
    System.out.println(
        """

            For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources

            to set up a load-balanced web service endpoint and explore
some ways to make it resilient

            against various kinds of failures.

            Some of the resources create by this demo are:
            \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
            \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.

```

```

        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
        """);

    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating and populating a DynamoDB table named " +
tableName);
    Database database = new Database();
    database.createTable(tableName, fileName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("""
        Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
        This script starts a Python web server defined in the `server.py`
script. The web server
        listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
        For demo purposes, this server is run as the root user. In
production, the best practice is to
        run a web server, such as Apache, with least-privileged credentials.

        The template also defines an IAM policy that each instance uses to
assume a role that grants
        permissions to access the DynamoDB recommendation table and Systems
Manager parameters
        that control the flow of the demo.
        """);

    LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
    templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println(

```

```
        "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
            At this point, you have EC2 instances created. Once each instance
starts, it listens for
            HTTP requests. You can see these instances in the console or
continue with the demo.
            Press Enter when you're ready to continue.
            """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
            Creating an Elastic Load Balancing target group and load balancer.
The target group
            defines how the load balancer connects to instances. The load
balancer provides a
            single endpoint where clients connect and dispatches requests to
instances in the group.
            """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
```



```

        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessful) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                        For this example to work, the default security group for
your default VPC must
                        allow access from this computer. You can either add it
automatically from this
                        example or add it yourself using the AWS Management
Console.
                        """);

                    System.out.println(
                        "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {

```

```

        System.out.println("No security group rule added.");
    }
}

} catch (AutoScalingException e) {
    e.printStackTrace();
}
} else if (wasSuccessful) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" + elbDnsName);
} else {
    System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
    System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
    System.out.println("you can successfully make a GET request to the load
balancer.");
}

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();
    System.out.println("Read the ssm_only_policy.json file");
    String ssmOnlyPolicy = readFileAsString(ssmJSON);

    System.out.println("Resetting parameters to starting values for demo.");
    paramHelper.reset();

    System.out.println(
        """"
                This part of the demonstration shows how to toggle
different parts of the system
                to create situations where the web service fails, and shows
how using a resilient
                architecture can keep the web service running in spite of
these failures.
        """);
}

```

```
        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
        """);
demoChoices(loadBalancer);

System.out.println(
    ""
        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
        """);
paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

System.out.println(
    ""
        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
        """);
demoChoices(loadBalancer);

System.out.println(
    ""
        Instead of failing when the recommendation service fails,
the web service can return a static response.
        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
        """);
paramHelper.put(paramHelper.failureResponse, "static");

System.out.println("""
    Now, sending a GET request to the load balancer endpoint returns a
static response.
    The service still reports as healthy because health checks are still
shallow.
    """);
demoChoices(loadBalancer);

System.out.println("Let's reinstate the recommendation service.");
paramHelper.put(paramHelper.tableName, paramHelper.dyntable);
```

```
System.out.println("""
    Let's also substitute bad credentials for one of the instances in
the target group so that it can't
    access the DynamoDB recommendation table. We will get an instance id
value.
    """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
AutoScaler autoScaler = new AutoScaler();

// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " + profileAssociationId);
System.out.println("Replacing the profile for instance " + badInstanceId
    + " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
    """
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
    """);

demoChoices(loadBalancer);

System.out.println("""
    Let's implement a deep health check. For this demo, a deep health
check tests whether
    the web service can access the DynamoDB table that it depends on for
recommendations. Note that
    the deep health check is only for ELB routing and not for Auto
Scaling instance health.
    This kind of deep health check is not recommended for Auto Scaling
instance health, because it
```

```
        risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
        """);

        System.out.println("""
            By implementing deep health checks, the load balancer can detect
when one of the instances is failing
            and take that instance out of rotation.
        """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
            Now, checking target health indicates that the instance with bad
credentials
            is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
            instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
            the load balancer takes unhealthy instances out of its rotation.
        """);

        demoChoices(loadBalancer);

        System.out.println(
            """
                Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                instance is to terminate it and let the auto scaler start a
new instance to replace it.
            """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
            Even while the instance is terminating and the new instance is
starting, sending a GET
            request to the web service continues to get a successful
recommendation response because
            the load balancer routes requests to the healthy instances. After
the replacement instance
            starts and reports as healthy, it is included in the load balancing
rotation.

            Note that terminating and replacing an instance typically takes
several minutes, during which time you
```

```
        can see the changing health check status until the new instance is
running and healthy.
        """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();
```

```
        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);

        // Display the JSON response
        BufferedReader reader = new BufferedReader(
            new
InputStreamReader(response.getEntity().getContent()));
        StringBuilder jsonResponse = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            jsonResponse.append(line);
        }
        reader.close();

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();
    }
    case 1 -> {
        System.out.println("\nChecking the health of load balancer
targets:\n");

        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
            Note that it can take a minute or two for the health
check to update

            after changes are made.
            """);
    }
}
```

```

        }
        case 2 -> {
            System.out.println("\n0kay, let's move on.");
            System.out.println("-".repeat(88));
            return; // Exit the method when choice is 2
        }
        default -> System.out.println("You must choose a value between
0-2. Please select again.");
    }

    } catch (java.util.InputMismatchException e) {
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}

```

Membuat kelas yang menggabungkan tindakan Penskalaan Otomatis dan Amazon EC2.

```

public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return iamClient;
    }
}

```



```
private SsmClient getSSMClient() {
    if (ssmClient == null) {
        ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ssmClient;
}

private Ec2Client getEc2Client() {
    if (ec2Client == null) {
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    System.out.format("Terminated instance %s.", instanceId);
}
}
```

```
/**
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
    software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
    .builder()
    .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
    .build();

    // Replace the IAM instance profile association for the EC2 instance.
    ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
    .builder()
    .iamInstanceProfile(iamInstanceProfile)
    .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
    .build();

    try {
        getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
        // Handle the response as needed.
    } catch (Ec2Exception e) {
        // Handle exceptions, log, or report the error.
        System.err.println("Error: " + e.getMessage());
    }

    System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
        newInstanceProfileName);
    TimeUnit.SECONDS.sleep(15);
    boolean instReady = false;
    int tries = 0;

    // Reboot after 60 seconds
```

```
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                    .instanceIds(instanceId)
                    .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
            .instanceIds(instanceId)
            .documentName("AWS-RunShellScript")
            .parameters(Collections.singletonMap("commands",
                Collections.singletonList("cd / && sudo python3 server.py
80")))
            .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(secGroupId)
```

```

        .cidrIp(ipAddress)
        .fromPort(Integer.parseInt(port))
        .build();

    getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
    System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
}

/**
 * Detaches a role from an instance profile, detaches policies from the role,
 * and deletes all the resources.
 */
public void deleteInstanceProfile(String roleName, String profileName) {
    try {
        software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
        .builder()
        .instanceProfileName(profileName)
        .build();

        GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
        String name = response.getInstanceProfile().getInstanceProfileName();
        System.out.println(name);

        RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .roleName(roleName)
        .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
        .roleName(roleName)

```

```
        .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
            .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(attachedPolicy.policyArn())
                .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");

    } catch (IamException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
        .autoScalingGroupName(groupName)
        .forceDelete(true)
        .build();

    getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}
```

```
/*
 * Verify the default security group of the specified VPC allows ingress from
 * this
 * computer. This can be done by allowing ingress from this computer's IP
 * address. In some situations, such as connecting from a corporate network, you
 * must instead specify a prefix list ID. You can also temporarily open the port
 * to
 * any IP address while running this example. If you do, be sure to remove
 * public
 * access when you're done.
 */
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
            .filters(filter, filter1)
            .build();

        DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
            .describeSecurityGroups(securityGroupsRequest);
        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " + secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " + ipPermission);
                }
            }
        }
    }
}
```

```

        for (IpRange ipRange : ipPermission.ipRanges()) {
            String cidrIp = ipRange.cidrIp();
            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                System.out.println(cidrIp + " is applicable");
                portIsOpen = true;
            }
        }

        if (!ipPermission.prefixListIds().isEmpty()) {
            System.out.println("Prefix lList is applicable");
            portIsOpen = true;
        }

        if (!portIsOpen) {
            System.out
                .println("The inbound rule does not appear to be
open to either this computer's IP,"
                    + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
        } else {
            break;
        }
    }
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/**
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {

```

```
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                .autoScalingGroupName(asGroupName)
                .targetGroupARNs(targetGroupARN)
                .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
```



```
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

    try {
        getAutoScalingClient().createAutoScalingGroup(groupRequest);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
    return zones;
}

public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
```

```
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();

    DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
        .filters(vpcFilter, azFilter, defaultForAZ)
        .build();

    DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
    subnets = response.subnets();
    return subnets;
}

// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}

// Gets data about the profile associated with an instance.
```

```

public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
    ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
    ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
    for (Policy policy : listPoliciesResponse.policies()) {
        if (policy.policyName().equals(policyName)) {
            // List the entities (users, groups, roles) that are attached to the
policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
        .builder()
        .policyArn(policy.arn())
        .build();
        ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
            .listEntitiesForPolicy(listEntitiesRequest);
        if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
            || !listEntitiesResponse.policyRoles().isEmpty()) {
            // Detach the policy from any entities it is attached to.
            DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
                .policyArn(policy.arn())
                .roleName(roleName) // Specify the name of the IAM role

```

```
        .build();

        getIAMClient().detachRolePolicy(detachPolicyRequest);
        System.out.println("Policy detached from entities.");
    }

    // Now, you can delete the policy.
    DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
        .policyArn(policy.arn())
        .build();

    getIAMClient().deletePolicy(deletePolicyRequest);
    System.out.println("Policy deleted successfully.");
    break;
}
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .roleName(roleName) // Remove the extra dot here
        .build();

    getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
    System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
}

// Delete the instance profile after removing all roles
DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
```

```

        .build();

        getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}

```

Membuat kelas yang menggabungkan tindakan Penyeimbangan Beban Elastis.

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
            .names(targetGroupName)
            .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
            .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
            .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }
}

```

```
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
```

```

        .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {
            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);
            if (statusCode == 200) {
                success = true;
            } else {
                retries--;
                System.out.println("Got connection error from load balancer
endpoint, retrying...");
                TimeUnit.SECONDS.sleep(15);
            }
        }
    } catch (org.apache.http.conn.HttpHostConnectException e) {
        System.out.println(e.getMessage());
    }

    System.out.println("Status.." + success);
    return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies

```

```
    * how
    * the load balancer forward requests to instances in the group and how instance
    * health is checked.
    */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
            .healthCheckPath("/healthcheck")
            .healthCheckTimeoutSeconds(5)
            .port(port)
            .vpcId(vpcId)
            .name(targetGroupName)
            .protocol(protocol)
            .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
```



```
        .scheme("internet-facing")
        .build();

    // Create and wait for the load balancer to become available.
    CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
    String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

    ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
    DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
        .loadBalancerArns(lbARN)
        .build();

    System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
    WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
        .waitUntilLoadBalancerAvailable(request);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Load Balancer " + lbName + " is available.");

    // Get the DNS name (endpoint) of the load balancer.
    String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
    System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

    // Create a listener for the load balance.
    Action action = Action.builder()
        .targetGroupArn(targetGroupARN)
        .type("forward")
        .build();

    CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
        .defaultActions(action)
        .port(port)
        .protocol(protocol)
        .defaultActions(action)
        .build();

    getLoadBalancerClient().createListener(listenerRequest);
```

```

        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}

```

Membuat kelas yang menggunakan DynamoDB untuk menyimulasikan layanan yang direkomendasikan.

```

public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;
        }
    }
}

```

```

    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " + e.getMessage());
    }
    return false;
}

/**
 * Creates a DynamoDB table to use a recommendation service. The table has a
 * hash key named 'MediaType' that defines the type of media recommended, such
 * as
 * Book or Movie, and a range key named 'ItemId' that, combined with the
 * MediaType,
 * forms a unique identifier for the recommended item.
 */
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build()
            )
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build()
            )
            .provisionedThroughput(
                ProvisionedThroughput.builder()

```

```

        .readCapacityUnits(5L)
        .writeCapacityUnits(5L)
        .build()

        .build();

        getDynamoDbClient().createTable(createTableRequest);
        System.out.println("Creating table " + tableName + "...");

        // Wait until the Amazon DynamoDB table is created.
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        WaiterResponse<DescribeTableResponse> waiterResponse =
        dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Table " + tableName + " created.");

        // Add records to the table.
        populateTable(fileName, tableName);
    }
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
    }
}

```

```

String title = currentNode.path("Title").path("S").asText();
String creator = currentNode.path("Creator").path("S").asText();

// Create a Recommendation object and set its properties.
Recommendation rec = new Recommendation();
rec.setMediaType(mediaType);
rec.setItemId(itemId);
rec.setTitle(title);
rec.setCreator(creator);

// Put the item into the DynamoDB table.
mappedTable.putItem(rec); // Add the Recommendation to the list.
}
System.out.println("Added all records to the " + tableName);
}
}

```

Membuat kelas yang menggabungkan tindakan Systems Manager.

```

public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")

```

```
        .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```


- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .

- [AttachLoadBalancerTargetGroups](#)
- [CreateAutoScalingGroup](#)
- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacelamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)

- [UpdateAutoScalingGroup](#)

Buat pengguna dan ambil peran


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 for Java 2.x

 Note

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

Buat fungsi yang membungkus tindakan pengguna IAM.

```
/*  
  To run this Java V2 code example, set up your development environment, including  
  your credentials.  
  
  For information, see this documentation topic:  
  
  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
  
  This example performs these operations:
```

1. Creates a user that has no permissions.
  2. Creates a role and policy that grants Amazon S3 permissions.
  3. Creates a role.
  4. Grants the user permissions.
  5. Gets temporary credentials by assuming the role. Creates an Amazon S3 Service client object with the temporary credentials.
  6. Deletes the resources.
- \*/

```
public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static final String PolicyDocument = "{" +
        "  \"Version\": \"2012-10-17\",\" +
        "  \"Statement\": [\" +
        "    {\" +
        "      \"Effect\": \"Allow\",\" +
        "      \"Action\": [\" +
        "        \"s3:*\"\" +
        "      ],\" +
        "      \"Resource\": \"*\\" +
        "    }\" +
        "  ]\" +
        "};

    public static String userArn;

    public static void main(String[] args) throws Exception {

        final String usage = ""

            Usage:
                <username> <policyName> <roleName> <roleSessionName>
<bucketName>\\s

            Where:
                username - The name of the IAM user to create.\\s
                policyName - The name of the policy to create.\\s
                roleName - The name of the role to create.\\s
                roleSessionName - The name of the session required for the
assumeRole operation.\\s
                bucketName - The name of the Amazon S3 bucket from which objects
are read.\\s

            """;
```



```
if (args.length != 5) {
    System.out.println(usage);
    System.exit(1);
}

String userName = args[0];
String policyName = args[1];
String roleName = args[2];
String roleSessionName = args[3];
String bucketName = args[4];

Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the AWS IAM example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 1. Create the IAM user.");
User createUser = createIAMUser(iam, userName);

System.out.println(DASHES);
userArn = createUser.arn();

AccessKey myKey = createIAMAccessKey(iam, userName);
String accessKey = myKey.accessKeyId();
String secretKey = myKey.secretAccessKey();
String assumeRolePolicyDocument = "{" +
    "\"Version\": \"2012-10-17\"," +
    "\"Statement\": [{" +
    "\"Effect\": \"Allow\"," +
    "\"Principal\": {" +
    "  \"AWS\": \"" + userArn + "\"" +
    "}," +
    "\"Action\": \"sts:AssumeRole\"" +
    "}]}" +
    "}";

System.out.println(assumeRolePolicyDocument);
System.out.println(userName + " was successfully created.");
```

```
System.out.println(DASHES);
System.out.println("2. Creates a policy.");
String polArn = createIAMPolicy(iam, policyName);
System.out.println("The policy " + polArn + " was successfully created.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Creates a role.");
TimeUnit.SECONDS.sleep(30);
String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
System.out.println(roleArn + " was successfully created.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Grants the user permissions.");
attachIAMRolePolicy(iam, roleName, polArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("*** Wait for 30 secs so the resource is available");
TimeUnit.SECONDS.sleep(30);
System.out.println("5. Gets temporary credentials by assuming the role.");
System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6 Getting ready to delete the AWS resources");
deleteKey(iam, userName, accessKey);
deleteRole(iam, roleName, polArn);
deleteIAMUser(iam, userName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("This IAM Scenario has successfully completed");
System.out.println(DASHES);
}

public static AccessKey createIAMAccessKey(IamClient iam, String user) {
    try {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(user)
            .build();
```

```
        CreateAccessKeyResponse response = iam.createAccessKey(request);
        return response.accessKey();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static User createIAMUser(IamClient iam, String username) {
    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();
        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        // Wait until the user is created.
        CreateUserResponse response = iam.createUser(request);
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static String createIAMRole(IamClient iam, String rolename, String json)
{

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(json)
```

```
        .description("Created using the AWS SDK for Java")
        .build();

    CreateRoleResponse response = iam.createRole(request);
    System.out.println("The ARN of the role is " + response.role().arn());
    return response.role().arn();

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static String createIAMPolicy(IamClient iam, String policyName) {
    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();
        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument).build();

        CreatePolicyResponse response = iam.createPolicy(request);
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
    try {
```

```
        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
        .roleName(roleName)
        .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
        String polArn;
        for (AttachedPolicy policy : attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn) == 0) {
                System.out.println(roleName + " policy is already attached to
this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(policyArn)
        .build();

        iam.attachRolePolicy(attachRequest);
        System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Invoke an Amazon S3 operation using the Assumed Role.
public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
        String keySecret) {

    // Use the creds of the new IAM user that was created in this code example.
    AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
    StsClient stsClient = StsClient.builder()
        .region(Region.US_EAST_1)
```

```
        .credentialsProvider(StaticCredentialsProvider.create(credentials))
        .build();

    try {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
            .roleArn(roleArn)
            .roleSessionName(roleSessionName)
            .build();

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        Credentials myCreds = roleResponse.credentials();
        String key = myCreds.accessKeyId();
        String secKey = myCreds.secretAccessKey();
        String secToken = myCreds.sessionToken();

        // List all objects in an Amazon S3 bucket using the temp creds
retrieved by
        // invoking assumeRole.
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .credentialsProvider(
StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
secToken)))
            .region(region)
            .build();

        System.out.println("Created a S3Client using temp credentials.");
        System.out.println("Listing objects in " + bucketName);
        ListObjectsRequest listObjects = ListObjectsRequest.builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.println("The name of the key is " + myValue.key());
            System.out.println("The owner is " + myValue.owner());
        }

    } catch (StsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}

public static void deleteRole(IamClient iam, String roleName, String polArn) {

    try {
        // First the policy needs to be detached.
        DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
            .policyArn(polArn)
            .roleName(roleName)
            .build();

        iam.detachRolePolicy(rolePolicyRequest);

        // Delete the policy.
        DeletePolicyRequest request = DeletePolicyRequest.builder()
            .policyArn(polArn)
            .build();

        iam.deletePolicy(request);
        System.out.println("*** Successfully deleted " + polArn);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteKey(IamClient iam, String username, String accessKey) {
    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
    }
}
```

```
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteIAMUser(IamClient iam, String userName) {
    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("*** Successfully deleted " + userName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)



- [PutUserPolicy](#)

## Bekerja dengan API Pembuat Kebijakan IAM

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat kebijakan IAM dengan menggunakan API berorientasi objek.
- Gunakan API Pembuat Kebijakan IAM dengan layanan IAM.

## SDK for Java 2.x

### Note

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

Contoh menggunakan impor berikut.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.policybuilder.iam.IamConditionOperator;
import software.amazon.awssdk.policybuilder.iam.IamEffect;
import software.amazon.awssdk.policybuilder.iam.IamPolicy;
import software.amazon.awssdk.policybuilder.iam.IamPolicyWriter;
import software.amazon.awssdk.policybuilder.iam.IamPrincipal;
import software.amazon.awssdk.policybuilder.iam.IamPrincipalType;
import software.amazon.awssdk.policybuilder.iam.IamResource;
import software.amazon.awssdk.policybuilder.iam.IamStatement;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyVersionResponse;
import software.amazon.awssdk.services.sts.StsClient;

import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;
```

Buat kebijakan berbasis waktu.

```
public String timeBasedPolicyExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addResource(IamResource.ALL)
            .addCondition(b1 -> b1

        .operator(IamConditionOperator.DATE_GREATER_THAN)

        .key("aws:CurrentTime")

        .value("2020-04-01T00:00:00Z"))
            .addCondition(b1 -> b1

        .operator(IamConditionOperator.DATE_LESS_THAN)

        .key("aws:CurrentTime")

        .value("2020-06-30T23:59:59Z")))
        .build();

    // Use an IamPolicyWriter to write out the JSON string to a more
readable
    // format.
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true)
        .build());
}
```

Buat kebijakan dengan beberapa kondisi.

```
public String multipleConditionsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addAction("dynamodb:BatchGetItem")
```

```

        .addAction("dynamodb:Query")
        .addAction("dynamodb:PutItem")
        .addAction("dynamodb:UpdateItem")
        .addAction("dynamodb>DeleteItem")

        .addAction("dynamodb:BatchWriteItem")

        .addResource("arn:aws:dynamodb:*:*:table/table-name")

        .addConditions(IamConditionOperator.STRING_EQUALS

        .addPrefix("ForAllValues:"),

        "dynamodb:Attributes",

        List.of("column-
name1", "column-name2", "column-name3"))

        .addCondition(b1 -> b1

        .operator(IamConditionOperator.STRING_EQUALS

        .addSuffix("IfExists"))

        .key("dynamodb>Select")

        .value("SPECIFIC_ATTRIBUTES"))))

        .build();

        return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
    }

```

Gunakan prinsipal dalam kebijakan.

```

public String specifyPrincipalsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.DENY)
            .addAction("s3:*")
            .addPrincipal(IamPrincipal.ALL)

        .addResource("arn:aws:s3::BUCKETNAME/*")

```

```

.addResource("arn:aws:s3:::BUCKETNAME")
                                .addCondition(b1 -> b1

.operator(IamConditionOperator.ARN_NOT_EQUALS)

.key("aws:PrincipalArn")

.value("arn:aws:iam::444455556666:user/user-name")))
                                .build();
return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
}

```

Izinkan akses lintas akun.

```

public String allowCrossAccountAccessExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addPrincipal(IamPrincipalType.AWS,
"111122223333")
            .addAction("s3:PutObject")
            .addResource("arn:aws:s3:::DOC-
EXAMPLE-BUCKET/*")
            .addCondition(b1 -> b1

.operator(IamConditionOperator.STRING_EQUALS)
                                .key("s3:x-amz-acl")
                                .value("bucket-
owner-full-control")))
        .build();
return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

Membangun dan meng-upload fileIamPolicy.

```

public String createAndUploadPolicyExample(IamClient iam, String accountID,
String policyName) {
    // Build the policy.

```

```

        IamPolicy policy = IamPolicy.builder() // 'version' defaults to
"2012-10-17".
                .addStatement(IamStatement.builder()
                        .effect(IamEffect.ALLOW)
                        .addAction("dynamodb:PutItem")
                        .addResource("arn:aws:dynamodb:us-
east-1:" + accountID
                                + ":table/
exampleTableName")
                        .build())
                .build();
        // Upload the policy.
        iam.createPolicy(r ->
r.policyName(policyName).policyDocument(policy.toJson()));
        return
policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
    }

```

Unduh dan kerjakan dengan `fileIamPolicy`.

```

    public String createNewBasedOnExistingPolicyExample(IamClient iam, String
accountID, String policyName,
                String newPolicyName) {

        String policyArn = "arn:aws:iam::" + accountID + ":policy/" +
policyName;
        GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

        String policyVersion =
getPolicyResponse.policy().defaultVersionId();
        GetPolicyVersionResponse getPolicyVersionResponse = iam
                .getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

        // Create an IamPolicy instance from the JSON string returned from
IAM.
        String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
                StandardCharsets.UTF_8);
        IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

```

```
        /*
         * All IamPolicy components are immutable, so use the copy method
that creates a
         * new instance that
         * can be altered in the same method call.
         *
         * Add the ability to get an item from DynamoDB as an additional
action.
         */
        IAMStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

        // Create a new statement that replaces the original statement.
        IAMPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

        // Upload the new policy. IAM now has both policies.
        iam.createPolicy(r -> r.policyName(newPolicyName)
            .policyDocument(newPolicy.toJson()));

        return
newPolicy.toJson(IAMPolicyWriter.builder().prettyPrint(true).build());
    }
}
```

- Untuk informasi selengkapnya, silakan lihat [Panduan Developer AWS SDK for Java 2.x](#).
- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [CreatePolicy](#)
  - [GetPolicy](#)
  - [GetPolicyVersion](#)

## AWS IoT contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with AWS IoT.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.


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

Memulai

Halo AWS IoT

Contoh kode berikut menunjukkan cara untuk mulai menggunakan AWS IoT.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
import java.util.List;

public class HelloIoT {
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllThings(iotClient);
    }

    public static void listAllThings( IotClient iotClient) {
        ListThingsRequest thingsRequest = ListThingsRequest.builder()
            .maxResults(10)
            .build();
```

```
ListThingsResponse response = iotClient.listThings(thingsRequest) ;
List<ThingAttribute> thingList = response.things();
for (ThingAttribute attribute : thingList) {
    System.out.println("Thing name: "+attribute.thingName());
    System.out.println("Thing ARN: "+attribute.thingArn());
}
}
```

- Untuk detail API, lihat [ListThings](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

### Lampirkan sertifikat

Contoh kode berikut menunjukkan cara melampirkan AWS IoT sertifikat.

### SDK for Java 2.x

#### Note

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

```
public static void attachCertificateToThing(IotClient iotClient, String
thingName, String certificateArn) {
    // Attach the certificate to the thing.
    AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
    .thingName(thingName)
    .principal(certificateArn)
    .build();
```



```
AttachThingPrincipalResponse attachResponse =
iotClient.attachThingPrincipal(principalRequest);

// Verify the attachment was successful.
if (attachResponse.sdkHttpResponse().isSuccessful()) {
    System.out.println("Certificate attached to Thing successfully.");

    // Print additional information about the Thing.
    describeThing(iotClient, thingName);
} else {
    System.err.println("Failed to attach certificate to Thing. HTTP Status
Code: " +
        attachResponse.sdkHttpResponse().statusCode());
}
}
```

- Untuk detail API, lihat [AttachThingPrincipal](#) di Referensi AWS SDK for Java 2.x API.

## Buat sertifikat

Contoh kode berikut menunjukkan cara membuat AWS IoT sertifikat.

## SDK for Java 2.x

### Note

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

```
public static String createCertificate(IotClient iotClient) {
    try {
        CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
        String certificatePem = response.certificatePem();
        String certificateArn = response.certificateArn();

        // Print the details.
        System.out.println("\nCertificate:");
```

```

        System.out.println(certificatePem);
        System.out.println("\nCertificate ARN:");
        System.out.println(certificateArn);
        return certificateArn;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

```

- Untuk detail API, lihat [CreateKeysAndCertificate](#) di Referensi AWS SDK for Java 2.x API.

Buat aturan

Contoh kode berikut menunjukkan cara membuat AWS IoT aturan.

SDK for Java 2.x

#### Note

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

```

public static void createIoTRule(IotClient iotClient, String roleARN, String
ruleName, String action) {
    try {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();
    }
}

```

```
// Create the topic rule payload.
TopicRulePayload topicRulePayload = TopicRulePayload.builder()
    .sql(sql)
    .actions(myAction)
    .build();

// Create the topic rule request.
CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
    .ruleName(ruleName)
    .topicRulePayload(topicRulePayload)
    .build();

// Create the rule.
iotClient.createTopicRule(topicRuleRequest);
System.out.println("IoT Rule created successfully.");


} catch (IotException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Untuk detail API, lihat [CreateTopicRule](#) di Referensi AWS SDK for Java 2.x API.

Buat sesuatu

Contoh kode berikut menunjukkan cara membuat AWS IoT sesuatu.

SDK for Java 2.x

 Note

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

```
public static void createIoTThing(IotClient iotClient, String thingName) {
    try {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
```

```

        .build();

        CreateThingResponse createThingResponse =
iotClient.createThing(createThingRequest);
        System.out.println(thingName + " was successfully created. The ARN value
is " + createThingResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Untuk detail API, lihat [CreateThing](#) di Referensi AWS SDK for Java 2.x API.

## Hapus sertifikat

Contoh kode berikut menunjukkan cara menghapus AWS IoT sertifikat.

SDK for Java 2.x

### Note

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

```

public static void deleteCertificate(IotClient iotClient, String
certificateArn ) {
    DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
        .certificateId(extractCertificateId(certificateArn))
        .build();

    iotClient.deleteCertificate(certificateProviderRequest);
    System.out.println(certificateArn + " was successfully deleted.");
}

```

- Untuk detail API, lihat [DeleteCertificate](#) di Referensi AWS SDK for Java 2.x API.

## Hapus sesuatu

Contoh kode berikut menunjukkan cara menghapus AWS IoT sesuatu.

SDK for Java 2.x

### Note

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

```
public static void deleteIoTThing(IotClient iotClient, String thingName) {
    try {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        iotClient.deleteThing(deleteThingRequest);
        System.out.println("Deleted Thing " + thingName);
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DeleteThing](#) di Referensi AWS SDK for Java 2.x API.

## Jelaskan sesuatu

Contoh kode berikut menunjukkan bagaimana mendeskripsikan AWS IoT sesuatu.

SDK for Java 2.x

### Note

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

```
private static void describeThing(IotClient iotClient, String thingName) {
    try {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build() ;

        // Print Thing details.
        DescribeThingResponse describeResponse =
iotClient.describeThing(thingRequest);
        System.out.println("Thing Details:");
        System.out.println("Thing Name: " + describeResponse.thingName());
        System.out.println("Thing ARN: " + describeResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DescribeThing](#) di Referensi AWS SDK for Java 2.x API.

## Lepaskan sertifikat

Contoh kode berikut menunjukkan cara melepaskan AWS IoT sertifikat.

### SDK for Java 2.x

#### Note

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

```
public static void detachThingPrincipal(IotClient iotClient, String thingName,
String certificateArn){
    try {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
```

```
        .build();

        iotClient.detachThingPrincipal(thingPrincipalRequest);
        System.out.println(certificateArn + " was successfully removed from "
+thingName);

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DetachThingPrincipal](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan informasi titik akhir

Contoh kode berikut menunjukkan cara mendapatkan informasi AWS IoT endpoint.

SDK for Java 2.x

#### Note

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

```
public static String describeEndpoint(IotClient iotClient) {
    try {
        DescribeEndpointResponse endpointResponse =
        iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

        // Get the endpoint URL.
        String endpointUrl = endpointResponse.endpointAddress();
        String exString = getValue(endpointUrl);
        String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

        System.out.println("Full Endpoint URL: " + fullEndpoint);
        return fullEndpoint;
    }
}
```

```
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}
```

- Untuk detail API, lihat [DescribeEndpoint](#) di Referensi AWS SDK for Java 2.x API.

## Cantumkan sertifikat Anda

Contoh kode berikut menunjukkan cara membuat daftar AWS IoT sertifikat Anda.

### SDK for Java 2.x

#### Note

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

```
public static void listCertificates(IotClient iotClient) {
    ListCertificatesResponse response = iotClient.listCertificates();
    List<Certificate> certList = response.certificates();
    for (Certificate cert : certList) {
        System.out.println("Cert id: " + cert.certificateId());
        System.out.println("Cert Arn: " + cert.certificateArn());
    }
}
```


- Untuk detail API, lihat [ListCertificates](#) di Referensi AWS SDK for Java 2.x API.

## Kueri indeks pencarian

Contoh kode berikut menunjukkan bagaimana untuk query indeks AWS IoT pencarian.



## SDK for Java 2.x

 Note

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

```
public static void searchThings(IotClient iotClient, String queryString){
    SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
        .queryString(queryString)
        .build();

    try {
        // Perform the search and get the result.
        SearchIndexResponse searchIndexResponse =
        iotClient.searchIndex(searchIndexRequest);

        // Process the result.
        if (searchIndexResponse.things().isEmpty()) {
            System.out.println("No things found.");
        } else {
            searchIndexResponse.things().forEach(thing ->
            System.out.println("Thing id found using search is " + thing.thingId()));
        }
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [SearchIndex](#) di Referensi AWS SDK for Java 2.x API.

Perbarui sesuatu

Contoh kode berikut menunjukkan cara memperbarui AWS IoT sesuatu.

## SDK for Java 2.x

 Note

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

```
public static void updateThing(IotClient iotClient, String thingName) {
    // Specify the new attribute values.
    String newLocation = "Office";
    String newFirmwareVersion = "v2.0";

    Map<String, String> attMap = new HashMap<>();
    attMap.put("location", newLocation);
    attMap.put("firmwareVersion", newFirmwareVersion);

    AttributePayload attributePayload = AttributePayload.builder()
        .attributes(attMap)
        .build();

    UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
        .thingName(thingName)
        .attributePayload(attributePayload)
        .build();

    try {
        // Update the IoT Thing attributes.
        iotClient.updateThing(updateThingRequest);
        System.out.println("Thing attributes updated successfully.");
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [UpdateThing](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

Bekerja dengan kasus penggunaan manajemen perangkat

Contoh kode berikut menunjukkan cara bekerja dengan kasus penggunaan manajemen AWS IoT perangkat menggunakan AWS IoT SDK

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.Action;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.AttributePayload;
import software.amazon.awssdk.services.iot.model.Certificate;
import software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointResponse;
import software.amazon.awssdk.services.iot.model.DescribeThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeThingResponse;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.IotException;
import software.amazon.awssdk.services.iot.model.ListCertificatesResponse;
import software.amazon.awssdk.services.iot.model.ListTopicRulesRequest;
import software.amazon.awssdk.services.iot.model.ListTopicRulesResponse;
import software.amazon.awssdk.services.iot.model.SearchIndexRequest;
import software.amazon.awssdk.services.iot.model.SearchIndexResponse;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
```

```
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iot.model.UpdateThingRequest;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowResponse;
import software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowRequest;
import java.net.URI;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Creates an AWS IoT Thing.
 * 2. Generate and attach a device certificate.
 * 3. Update an AWS IoT Thing with Attributes.
 * 4. Get an AWS IoT Endpoint.
 * 5. List your certificates.
 * 6. Updates the shadow for the specified thing..
 * 7. Write out the state information, in JSON format
 * 8. Creates a rule
 * 9. List rules
 * 10. Search things
 * 11. Detach and delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String TOPIC = "your-iot-topic";
    public static void main(String[] args) {
        final String usage =
            ""
```

```

        Usage:
            <roleARN> <snsAction>

        Where:
            roleARN - The ARN of an IAM role that has permission to work
with AWS IOT.
            snsAction - An ARN of an SNS topic.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String thingName;
    String ruleName;
    String roleARN = args[0];
    String snsAction = args[1];
    Scanner scanner = new Scanner(System.in);
    IotClient iotClient = IotClient.builder()
        .region(Region.US_EAST_1)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the AWS IoT example workflow.");
    System.out.println("""
        This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service. The program guides you through a series of
steps,
        including creating an IoT Thing, generating a device certificate,
updating the Thing with attributes, and so on.
        It utilizes the AWS SDK for Java V2 and incorporates functionality for
creating and managing IoT Things, certificates, rules,
        shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
        developers working with AWS IoT in a Java environment.

        """);
    System.out.print("Press Enter to continue...");
    scanner.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Create an AWS IoT Thing.");

```

```
System.out.println("""
    An AWS IoT Thing represents a virtual entity in the AWS IoT service that
    can be associated with a physical device.
    """);
// Prompt the user for input.
System.out.print("Enter Thing name: ");
thingName = scanner.nextLine();
createIoTThing(iotClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Generate a device certificate.");
System.out.println("""
    A device certificate performs a role in securing the communication
    between devices (Things) and the AWS IoT platform.
    """);

System.out.print("Do you want to create a certificate for " +thingName +"?
(y/n)");
String certAns = scanner.nextLine();
String certificateArn="" ;
if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
    certificateArn = createCertificate(iotClient);
    System.out.println("Attach the certificate to the AWS IoT Thing.");
    attachCertificateToThing(iotClient, thingName, certificateArn);
} else {
    System.out.println("A device certificate was not created.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Update an AWS IoT Thing with Attributes.");
System.out.println("""
    IoT Thing attributes, represented as key-value pairs, offer a pivotal
    advantage in facilitating efficient data
    management and retrieval within the AWS IoT ecosystem.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
updateThing(iotClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
System.out.println("4. Return a unique endpoint specific to the Amazon Web
Services account.");
System.out.println("""
    An IoT Endpoint refers to a specific URL or Uniform Resource Locator
that serves as the entry point for communication between IoT devices and the AWS
IoT service.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
String endpointUrl = describeEndpoint(iotClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. List your AWS IoT certificates");
System.out.print("Press Enter to continue...");
scanner.nextLine();
if (certificateArn.length() > 0) {
    listCertificates(iotClient);
} else {
    System.out.println("You did not create a certificates. Skipping this
step.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
System.out.println("""
    A Thing Shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
    of a physical device or thing. The Thing Shadow allows you to
synchronize and control the state of a device between
    the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a Thing Shadow.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
IotDataPlaneClient iotPlaneClient = IotDataPlaneClient.builder()
    .region(Region.US_EAST_1)
    .endpointOverride(URI.create(endpointUrl))
    .build();

updateShadowThing(iotPlaneClient, thingName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("7. Write out the state information, in JSON format.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
getPayload(iotPlaneClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Creates a rule");
System.out.println("""
Creates a rule that is an administrator-level action.
Any user who has permission to create rules will be able to access data
processed by the rule.
""");
System.out.print("Enter Rule name: ");
ruleName = scanner.nextLine();
createIoTRule(iotClient, roleARN, ruleName, snsAction);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. List your rules.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
listIoTRules(iotClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Search things using the Thing name.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
String queryString = "thingName:"+thingName ;
searchThings(iotClient, queryString);
System.out.println(DASHES);

System.out.println(DASHES);
if (certificateArn.length() > 0) {
    System.out.print("Do you want to detach and delete the certificate for "
+thingName +"? (y/n)");
    String delAns = scanner.nextLine();
    if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
        System.out.println("11. You selected to detach amd delete the
certificate.");
        System.out.print("Press Enter to continue...");
```



```
        scanner.nextLine();
        detachThingPrincipal(iotClient, thingName, certificateArn);
        deleteCertificate(iotClient, certificateArn);
    } else {
        System.out.println("11. You selected not to delete the
certificate.");
    }
    } else {
        System.out.println("11. You did not create a certificate so there is
nothing to delete.");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("12. Delete the AWS IoT Thing.");
    System.out.print("Do you want to delete the IoT Thing? (y/n)");
    String delAns = scanner.nextLine();
    if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
        deleteIoTThing(iotClient, thingName);
    } else {
        System.out.println("The IoT Thing was not deleted.");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The AWS IoT workflow has successfully completed.");
    System.out.println(DASHES);
}

public static void listCertificates(IotClient iotClient) {
    ListCertificatesResponse response = iotClient.listCertificates();
    List<Certificate> certList = response.certificates();
    for (Certificate cert : certList) {
        System.out.println("Cert id: " + cert.certificateId());
        System.out.println("Cert Arn: " + cert.certificateArn());
    }
}

public static void listIoTRules(IotClient iotClient) {
    try {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        ListTopicRulesResponse listTopicRulesResponse =
iotClient.listTopicRules(listTopicRulesRequest);
```

```
        System.out.println("List of IoT Rules:");
        List<TopicRuleListItem> ruleList = listTopicRulesResponse.rules();
        for (TopicRuleListItem rule : ruleList) {
            System.out.println("Rule Name: " + rule.ruleName());
            System.out.println("Rule ARN: " + rule.ruleArn());
            System.out.println("-----");
        }

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createIoTRule(IotClient iotClient, String roleARN, String
ruleName, String action) {
    try {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        // Create the rule.
        iotClient.createTopicRule(topicRuleRequest);
        System.out.println("IoT Rule created successfully.");
    }
}
```

```
    } catch (IotException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getPayload(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        GetThingShadowResponse getThingShadowResponse =
iotPlaneClient.getThingShadow(getThingShadowRequest);

        // Extracting payload from response.
        SdkBytes payload = getThingShadowResponse.payload();
        String payloadString = payload.asUtf8String();
        System.out.println("Received Shadow Data: " + payloadString);

    } catch (IotException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateShadowThing(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}\"";
        SdkBytes data= SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8 );
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        // Update Thing Shadow.
```

```
        iotPlaneClient.updateThingShadow(updateThingShadowRequest);
        System.out.println("Thing Shadow updated successfully.");

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateThing(IotClient iotClient, String thingName) {
    // Specify the new attribute values.
    String newLocation = "Office";
    String newFirmwareVersion = "v2.0";

    Map<String, String> attMap = new HashMap<>();
    attMap.put("location", newLocation);
    attMap.put("firmwareVersion", newFirmwareVersion);

    AttributePayload attributePayload = AttributePayload.builder()
        .attributes(attMap)
        .build();

    UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
        .thingName(thingName)
        .attributePayload(attributePayload)
        .build();

    try {
        // Update the IoT Thing attributes.
        iotClient.updateThing(updateThingRequest);
        System.out.println("Thing attributes updated successfully.");

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String describeEndpoint(IotClient iotClient) {
    try {
        DescribeEndpointResponse endpointResponse =
        iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

        // Get the endpoint URL.
```

```
        String endpointUrl = endpointResponse.endpointAddress();
        String exString = getValue(endpointUrl);
        String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

        System.out.println("Full Endpoint URL: " + fullEndpoint);
        return fullEndpoint;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}

public static void detachThingPrincipal(IotClient iotClient, String thingName,
String certificateArn){
    try {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        iotClient.detachThingPrincipal(thingPrincipalRequest);
        System.out.println(certificateArn + " was successfully removed from "
+thingName);

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteCertificate(IotClient iotClient, String
certificateArn ) {
    DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
        .certificateId(extractCertificateId(certificateArn))
        .build();

    iotClient.deleteCertificate(certificateProviderRequest);
    System.out.println(certificateArn + " was successfully deleted.");
}
```

```
// Get the cert Id from the Cert ARN value.
private static String extractCertificateId(String certificateArn) {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
    String[] arnParts = certificateArn.split(":");
    String certificateIdPart = arnParts[arnParts.length - 1];
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1);
}

public static String createCertificate(IotClient iotClient) {
    try {
        CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
        String certificatePem = response.certificatePem();
        String certificateArn = response.certificateArn();

        // Print the details.
        System.out.println("\nCertificate:");
        System.out.println(certificatePem);
        System.out.println("\nCertificate ARN:");
        System.out.println(certificateArn);
        return certificateArn;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

public static void attachCertificateToThing(IotClient iotClient, String
thingName, String certificateArn) {
    // Attach the certificate to the thing.
    AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
        .thingName(thingName)
        .principal(certificateArn)
        .build();

    AttachThingPrincipalResponse attachResponse =
iotClient.attachThingPrincipal(principalRequest);

    // Verify the attachment was successful.
```

```
        if (attachResponse.sdkHttpResponse().isSuccessful()) {
            System.out.println("Certificate attached to Thing successfully.");

            // Print additional information about the Thing.
            describeThing(iotClient, thingName);
        } else {
            System.err.println("Failed to attach certificate to Thing. HTTP Status
Code: " +
                attachResponse.sdkHttpResponse().statusCode());
        }
    }

    private static void describeThing(IotClient iotClient, String thingName) {
        try {
            DescribeThingRequest thingRequest = DescribeThingRequest.builder()
                .thingName(thingName)
                .build();

            // Print Thing details.
            DescribeThingResponse describeResponse =
iotClient.describeThing(thingRequest);
            System.out.println("Thing Details:");
            System.out.println("Thing Name: " + describeResponse.thingName());
            System.out.println("Thing ARN: " + describeResponse.thingArn());

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteIoTThing(IotClient iotClient, String thingName) {
        try {
            DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
                .thingName(thingName)
                .build();

            iotClient.deleteThing(deleteThingRequest);
            System.out.println("Deleted Thing " + thingName);

        } catch (IotException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
}

public static void createIoTThing(IotClient iotClient, String thingName) {
    try {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CreateThingResponse createThingResponse =
iotClient.createThing(createThingRequest);
        System.out.println(thingName + " was successfully created. The ARN value
is " + createThingResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

private static String getValue(String input) {
    // Define a regular expression pattern for extracting the subdomain.
    Pattern pattern = Pattern.compile("^(.*)\\.\\.iot\\.\\.us-east-1\\.\\.amazonaws\\.
\\.com");

    // Match the pattern against the input string.
    Matcher matcher = pattern.matcher(input);

    // Check if a match is found.
    if (matcher.find()) {
        // Extract the subdomain from the first capturing group.
        String subdomain = matcher.group(1);
        System.out.println("Extracted subdomain: " + subdomain);
        return subdomain ;
    } else {
        System.out.println("No match found");
    }
    return "" ;
}

public static void searchThings(IotClient iotClient, String queryString){
    SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
        .queryString(queryString)
        .build();
```



```
    try {
        // Perform the search and get the result.
        SearchIndexResponse searchIndexResponse =
        iotClient.searchIndex(searchIndexRequest);

        // Process the result.
        if (searchIndexResponse.things().isEmpty()) {
            System.out.println("No things found.");
        } else {
            searchIndexResponse.things().forEach(thing ->
            System.out.println("Thing id found using search is " + thing.thingId()));
        }
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

## AWS IoT data contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with AWS IoT data.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

## Tindakan

### Dapatkan bayangannya

Contoh kode berikut menunjukkan bagaimana untuk mendapatkan bayangan untuk AWS IoT sesuatu.

### SDK for Java 2.x

#### Note

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

```
public static void getPayload(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        GetThingShadowResponse getThingShadowResponse =
iotPlaneClient.getThingShadow(getThingShadowRequest);

        // Extracting payload from response.
        SdkBytes payload = getThingShadowResponse.payload();
        String payloadString = payload.asUtf8String();
        System.out.println("Received Shadow Data: " + payloadString);

    } catch (IotException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [GetThingShadow](#) di Referensi AWS SDK for Java 2.x API.

## Perbarui bayangan

Contoh kode berikut menunjukkan cara memperbarui bayangan untuk suatu AWS IoT hal.

SDK for Java 2.x

### Note

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

```
public static void updateShadowThing(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
        \"humidity\":50}}}\"";
        SdkBytes data= SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8 );
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        // Update Thing Shadow.
        iotPlaneClient.updateThingShadow(updateThingShadowRequest);
        System.out.println("Thing Shadow updated successfully.");

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [UpdateThingShadow](#) di Referensi AWS SDK for Java 2.x API.

## Contoh Amazon Keyspaces menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with Amazon Keyspaces.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Memulai

Halo Amazon Keyspaces

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon Keyspaces.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.KeyspaceSummary;
import software.amazon.awssdk.services.keyspaces.model.KeyspacesException;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesRequest;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesResponse;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class HelloKeyspaces {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
            .region(region)
            .build();

        listKeyspaces(keyClient);
    }

    public static void listKeyspaces(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
                .maxResults(10)
                .build();

            ListKeyspacesResponse response =
keyClient.listKeyspaces(keyspacesRequest);
            List<KeyspaceSummary> keyspaces = response.keyspaces();
            for (KeyspaceSummary keyspace : keyspaces) {
                System.out.println("The name of the keyspace is " +
keyspace.keyspaceName());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListKeyspaces](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

Buat ruang kunci

Contoh kode berikut menunjukkan cara membuat keyspace Amazon Keyspaces.

SDK for Java 2.x

### Note

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

```
public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
        System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [CreateKeyspace](#) di Referensi AWS SDK for Java 2.x API.

Buat tabel

Contoh kode berikut menunjukkan cara membuat tabel Amazon Keyspaces.

## SDK for Java 2.x

 Note

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

```
public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        // Set the columns.
        ColumnDefinition defTitle = ColumnDefinition.builder()
            .name("title")
            .type("text")
            .build();

        ColumnDefinition defYear = ColumnDefinition.builder()
            .name("year")
            .type("int")
            .build();

        ColumnDefinition defReleaseDate = ColumnDefinition.builder()
            .name("release_date")
            .type("timestamp")
            .build();

        ColumnDefinition defPlot = ColumnDefinition.builder()
            .name("plot")
            .type("text")
            .build();

        List<ColumnDefinition> collist = new ArrayList<>();
        collist.add(defTitle);
        collist.add(defYear);
        collist.add(defReleaseDate);
        collist.add(defPlot);

        // Set the keys.
        PartitionKey yearKey = PartitionKey.builder()
            .name("year")
            .build();
```

```
PartitionKey titleKey = PartitionKey.builder()
    .name("title")
    .build();

List<PartitionKey> keyList = new ArrayList<>();
keyList.add(yearKey);
keyList.add(titleKey);

SchemaDefinition schemaDefinition = SchemaDefinition.builder()
    .partitionKeys(keyList)
    .allColumns(colList)
    .build();

PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
    .status(PointInTimeRecoveryStatus.ENABLED)
    .build();

CreateTableRequest tableRequest = CreateTableRequest.builder()
    .keyspaceName(keySpace)
    .tableName(tableName)
    .schemaDefinition(schemaDefinition)
    .pointInTimeRecovery(timeRecovery)
    .build();

CreateTableResponse response = keyClient.createTable(tableRequest);
System.out.println("The table ARN is " + response.resourceArn());

} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Untuk detail API, lihat [CreateTable](#) di Referensi AWS SDK for Java 2.x API.

## Hapus ruang kunci

Contoh kode berikut menunjukkan cara menghapus keyspace Amazon Keyspaces.



## SDK for Java 2.x

 Note

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

```
public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        keyClient.deleteKeyspace(deleteKeyspaceRequest);


    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DeleteKeyspace](#) di Referensi AWS SDK for Java 2.x API.

## Hapus tabel

Contoh kode berikut menunjukkan cara menghapus tabel Amazon Keyspaces.

## SDK for Java 2.x

 Note

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

```
public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
    try {
        DeleteTableRequest tableRequest = DeleteTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        keyClient.deleteTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DeleteTable](#) di Referensi AWS SDK for Java 2.x API.

## Mendapatkan data tentang keyspace

Contoh kode berikut menunjukkan cara mendapatkan data tentang keyspace Amazon Keyspaces.

### SDK for Java 2.x

#### Note

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

```
public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
        String name = response.keyspaceName();
        System.out.println("The " + name + " KeySpace is ready");
    }
}
```


```
    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [GetKeyspace](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan data tentang tabel

Contoh kode berikut menunjukkan cara mendapatkan data tentang tabel Amazon Keyspaces.

SDK for Java 2.x

 Note

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

```
public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
    throws InterruptedException {
    try {
        boolean tableStatus = false;
        String status;
        GetTableResponse response = null;
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        while (!tableStatus) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println(". The table status is " + status);

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true;
            }
        }
    }
}
```

```

        Thread.sleep(500);
    }

    List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
    for (ColumnDefinition def : cols) {
        System.out.println("The column name is " + def.name());
        System.out.println("The column type is " + def.type());
    }

} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

```

- Untuk detail API, lihat [GetTable](#) di Referensi AWS SDK for Java 2.x API.

## Daftar ruang kunci

Contoh kode berikut menunjukkan cara mencantumkan keyspaces Amazon Keyspaces.

## SDK for Java 2.x

### Note

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

```

public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
    try {
        ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
            .maxResults(10)
            .build();

        ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
        listRes.stream()
            .flatMap(r -> r.keyspaces().stream())
            .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));
    }
}

```

```
    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ListKeyspaces](#) di Referensi AWS SDK for Java 2.x API.

## Daftar tabel di ruang kunci

Contoh kode berikut menunjukkan cara membuat daftar tabel Amazon Keyspaces di keyspace.

SDK for Java 2.x

### Note

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

```
public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
    try {
        ListTablesRequest tablesRequest = ListTablesRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
        listRes.stream()
            .flatMap(r -> r.tables().stream())
            .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                " Table name: " + content.tableName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ListTables](#) di Referensi AWS SDK for Java 2.x API.

Kembalikan tabel ke titik waktu

Contoh kode berikut menunjukkan cara mengembalikan tabel Amazon Keyspaces ke titik waktu.

SDK for Java 2.x

#### Note

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

```
public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
    ZonedDateTime utc) {
    try {
        Instant myTime = utc.toInstant();
        RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
            .restoreTimestamp(myTime)
            .sourceTableName("Movie")
            .targetKeyspaceName(keyspaceName)
            .targetTableName("MovieRestore")
            .sourceKeyspaceName(keyspaceName)
            .build();

        RestoreTableResponse response =
            keyClient.restoreTable(restoreTableRequest);
        System.out.println("The ARN of the restored table is " +
            response.restoredTableARN());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [RestoreTable](#) di Referensi AWS SDK for Java 2.x API.

## Perbarui tabel

Contoh kode berikut menunjukkan cara memperbarui tabel Amazon Keyspaces.

### SDK for Java 2.x

#### Note

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

```
public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        ColumnDefinition def = ColumnDefinition.builder()
            .name("watched")
            .type("boolean")
            .build();

        UpdateTableRequest tableRequest = UpdateTableRequest.builder()
            .keyspaceName(keySpace)
            .tableName(tableName)
            .addColumnns(def)
            .build();

        keyClient.updateTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [UpdateTable](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

Memulai dengan keyspaces dan tabel

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- 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 for Java 2.x

### Note

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

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Before running this Java code example, you must create a
 * Java keystore (JKS) file and place it in your project's resources folder.
 *
 * This file is a secure file format used to hold certificate information for
 * Java applications. This is required to make a connection to Amazon Keyspaces.
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Create a keyspace.
 * 2. Check for keyspace existence.
 * 3. List keyspaces using a paginator.
 * 4. Create a table with a simple movie data schema and enable point-in-time
 * recovery.
 * 5. Check for the table to be in an Active state.
```



- \* 6. List all tables in the keyspace.
- \* 7. Use a Cassandra driver to insert some records into the Movie table.
- \* 8. Get all records from the Movie table.
- \* 9. Get a specific Movie.
- \* 10. Get a UTC timestamp for the current time.
- \* 11. Update the table schema to add a 'watched' Boolean column.
- \* 12. Update an item as watched.
- \* 13. Query for items with watched = True.
- \* 14. Restore the table back to the previous state using the timestamp.
- \* 15. Check for completion of the restore action.
- \* 16. Delete the table.
- \* 17. Confirm that both tables are deleted.
- \* 18. Delete the keyspace.
- \*/

```
public class ScenarioKeyspaces {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    /*
     * Usage:
     * fileName - The name of the JSON file that contains movie data. (Get this file
     * from the GitHub repo at resources/sample_file.)
     * keyspaceName - The name of the keyspace to create.
     */
    public static void main(String[] args) throws InterruptedException, IOException
    {
        String fileName = "<Replace with the JSON file that contains movie data>";
        String keyspaceName = "<Replace with the name of the keyspace to create>";
        String titleUpdate = "The Family";
        int yearUpdate = 2013;
        String tableName = "Movie";
        String tableNameRestore = "MovieRestore";
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
            .region(region)
            .build();

        DriverConfigLoader loader =
        DriverConfigLoader.fromClasspath("application.conf");
        CqlSession session = CqlSession.builder()
            .withConfigLoader(loader)
            .build();

        System.out.println(DASHES);
    }
}
```

```
System.out.println("Welcome to the Amazon Keyspaces example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create a keyspace.");
createKeySpace(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
Thread.sleep(5000);
System.out.println("2. Check for keyspace existence.");
checkKeyspaceExistence(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. List keyspaces using a paginator.");
listKeyspacesPaginator(keyClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Create a table with a simple movie data schema and
enable point-in-time recovery.");
createTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Check for the table to be in an Active state.");
Thread.sleep(6000);
checkTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. List all tables in the keyspace.");
listTables(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Use a Cassandra driver to insert some records into
the Movie table.");
Thread.sleep(6000);
loadData(session, fileName, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
System.out.println("8. Get all records from the Movie table.");
getMovieData(session, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Get a specific Movie.");
getSpecificMovie(session, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get a UTC timestamp for the current time.");
ZonedDateTime utc = ZonedDateTime.now(ZoneOffset.UTC);
System.out.println("DATETIME = " + Date.from(utc.toInstant()));
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Update the table schema to add a watched Boolean
column.");
updateTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Update an item as watched.");
Thread.sleep(10000); // Wait 10 secs for the update.
updateRecord(session, keyspaceName, titleUpdate, yearUpdate);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Query for items with watched = True.");
getWatchedData(session, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Restore the table back to the previous state using
the timestamp.");
System.out.println("Note that the restore operation can take up to 20
minutes.");
restoreTable(keyClient, keyspaceName, utc);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Check for completion of the restore action.");
Thread.sleep(5000);
checkRestoredTable(keyClient, keyspaceName, "MovieRestore");
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete both tables.");
        deleteTable(keyClient, keyspaceName, tableName);
        deleteTable(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Confirm that both tables are deleted.");
        checkTableDelete(keyClient, keyspaceName, tableName);
        checkTableDelete(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Delete the keyspace.");
        deleteKeyspace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The scenario has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                .keyspaceName(keyspaceName)
                .build();

            keyClient.deleteKeyspace(deleteKeyspaceRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkTableDelete(KeyspacesClient keyClient, String
keyspaceName, String tableName)
        throws InterruptedException {
        try {
```

```
String status;
GetTableResponse response;
GetTableRequest tableRequest = GetTableRequest.builder()
    .keyspaceName(keyspaceName)
    .tableName(tableName)
    .build();

// Keep looping until table cannot be found and a
ResourceNotFoundException is
// thrown.
while (true) {
    response = keyClient.getTable(tableRequest);
    status = response.statusAsString();
    System.out.println(". The table status is " + status);
    Thread.sleep(500);
}

} catch (ResourceNotFoundException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}
System.out.println("The table is deleted");
}

public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
    try {
        DeleteTableRequest tableRequest = DeleteTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        keyClient.deleteTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkRestoredTable(KeyspacesClient keyClient, String
keyspaceName, String tableName)
    throws InterruptedException {
    try {
        boolean tableStatus = false;
```

```
String status;
GetTableResponse response = null;
GetTableRequest tableRequest = GetTableRequest.builder()
    .keyspaceName(keyspaceName)
    .tableName(tableName)
    .build();

while (!tableStatus) {
    response = keyClient.getTable(tableRequest);
    status = response.statusAsString();
    System.out.println("The table status is " + status);

    if (status.compareTo("ACTIVE") == 0) {
        tableStatus = true;
    }
    Thread.sleep(500);
}

List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
for (ColumnDefinition def : cols) {
    System.out.println("The column name is " + def.name());
    System.out.println("The column type is " + def.type());
}

} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

}

public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
    ZonedDateTime utc) {
    try {
        Instant myTime = utc.toInstant();
        RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
            .restoreTimestamp(myTime)
            .sourceTableName("Movie")
            .targetKeyspaceName(keyspaceName)
            .targetTableName("MovieRestore")
            .sourceKeyspaceName(keyspaceName)
            .build();

        RestoreTableResponse response =
            keyClient.restoreTable(restoreTableRequest);
    }
}
```

```
        System.out.println("The ARN of the restored table is " +
response.restoredTableARN());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getWatchedData(CqlSession session, String keyspaceName) {
    ResultSet resultSet = session
        .execute("SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE
watched = true ALLOW FILTERING;");
    resultSet.forEach(item -> {
        System.out.println("The Movie title is " + item.getString("title"));
        System.out.println("The Movie year is " + item.getInt("year"));
        System.out.println("The plot is " + item.getString("plot"));
    });
}

public static void updateRecord(CqlSession session, String keySpace, String
titleUpdate, int yearUpdate) {
    String sqlStatement = "UPDATE \"" + keySpace
        + "\".\"Movie\" SET watched=true WHERE title = :k0 AND year = :k1;";
    BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
    builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
    PreparedStatement preparedStatement = session.prepare(sqlStatement);
    builder.addStatement(preparedStatement.boundStatementBuilder()
        .setString("k0", titleUpdate)
        .setInt("k1", yearUpdate)
        .build());

    BatchStatement batchStatement = builder.build();
    session.execute(batchStatement);
}

public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        ColumnDefinition def = ColumnDefinition.builder()
            .name("watched")
            .type("boolean")
            .build();
```

```

        UpdateTableRequest tableRequest = UpdateTableRequest.builder()
            .keyspaceName(keySpace)
            .tableName(tableName)
            .addColumnns(def)
            .build();

        keyClient.updateTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getSpecificMovie(CqlSession session, String keyspaceName) {
    ResultSet resultSet = session.execute(
        "SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE title = 'The
Family' ALLOW FILTERING ;");
    resultSet.forEach(item -> {
        System.out.println("The Movie title is " + item.getString("title"));
        System.out.println("The Movie year is " + item.getInt("year"));
        System.out.println("The plot is " + item.getString("plot"));
    });
}

// Get records from the Movie table.
public static void getMovieData(CqlSession session, String keyspaceName) {
    ResultSet resultSet = session.execute("SELECT * FROM \"" + keyspaceName +
    "\".\"Movie\";");
    resultSet.forEach(item -> {
        System.out.println("The Movie title is " + item.getString("title"));
        System.out.println("The Movie year is " + item.getInt("year"));
        System.out.println("The plot is " + item.getString("plot"));
    });
}

// Load data into the table.
public static void loadData(CqlSession session, String fileName, String
keySpace) throws IOException {
    String sqlStatement = "INSERT INTO \"" + keySpace + "\".\"Movie\" (title,
year, plot) values (:k0, :k1, :k2)";
    JsonParser parser = new JsonFactory().createParser(new File(fileName));

```



```
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {

        // Add 20 movies to the table.
        if (t == 20)
            break;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String plot = currentNode.path("info").path("plot").toString();

        // Insert the data into the Amazon Keyspaces table.
        BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
        PreparedStatement preparedStatement = session.prepare(sqlStatement);
        builder.addStatement(preparedStatement.boundStatementBuilder()
            .setString("k0", title)
            .setInt("k1", year)
            .setString("k2", plot)
            .build());

        BatchStatement batchStatement = builder.build();
        session.execute(batchStatement);
        t++;
    }

    System.out.println("You have added " + t + " records successfully!");
}

public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
    try {
        ListTablesRequest tablesRequest = ListTablesRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
        listRes.stream()
    }
}
```

```

        .flatMap(r -> r.tables().stream())
        .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
            " Table name: " + content.tableName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
    throws InterruptedException {
    try {
        boolean tableStatus = false;
        String status;
        GetTableResponse response = null;
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        while (!tableStatus) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println(". The table status is " + status);

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true;
            }
            Thread.sleep(500);
        }

        List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
        for (ColumnDefinition def : cols) {
            System.out.println("The column name is " + def.name());
            System.out.println("The column type is " + def.type());
        }

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

```
}

public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        // Set the columns.
        ColumnDefinition defTitle = ColumnDefinition.builder()
            .name("title")
            .type("text")
            .build();

        ColumnDefinition defYear = ColumnDefinition.builder()
            .name("year")
            .type("int")
            .build();

        ColumnDefinition defReleaseDate = ColumnDefinition.builder()
            .name("release_date")
            .type("timestamp")
            .build();

        ColumnDefinition defPlot = ColumnDefinition.builder()
            .name("plot")
            .type("text")
            .build();

        List<ColumnDefinition> collist = new ArrayList<>();
        collist.add(defTitle);
        collist.add(defYear);
        collist.add(defReleaseDate);
        collist.add(defPlot);

        // Set the keys.
        PartitionKey yearKey = PartitionKey.builder()
            .name("year")
            .build();

        PartitionKey titleKey = PartitionKey.builder()
            .name("title")
            .build();

        List<PartitionKey> keyList = new ArrayList<>();
        keyList.add(yearKey);
        keyList.add(titleKey);
    }
}
```

```
SchemaDefinition schemaDefinition = SchemaDefinition.builder()
    .partitionKeys(keyList)
    .allColumns(colList)
    .build();

PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
    .status(PointInTimeRecoveryStatus.ENABLED)
    .build();

CreateTableRequest tableRequest = CreateTableRequest.builder()
    .keyspaceName(keySpace)
    .tableName(tableName)
    .schemaDefinition(schemaDefinition)
    .pointInTimeRecovery(timeRecovery)
    .build();

CreateTableResponse response = keyClient.createTable(tableRequest);
System.out.println("The table ARN is " + response.resourceArn());

} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
    try {
        ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
            .maxResults(10)
            .build();

        ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
        listRes.stream()
            .flatMap(r -> r.keyspaces().stream())
            .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

```
public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
        String name = response.keyspaceName();
        System.out.println("The " + name + " KeySpace is ready");

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
        System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [CreateKeyspace](#)
  - [CreateTable](#)
  - [DeleteKeyspace](#)

- [DeleteTable](#)
- [GetKeyspace](#)
- [GetTable](#)
- [ListKeyspaces](#)
- [ListTables](#)
- [RestoreTable](#)
- [UpdateTable](#)

## Contoh Kinesis menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x Kinesis with.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)
- [Contoh nirserver](#)

## Tindakan

Buat aliran

Contoh kode berikut menunjukkan cara membuat aliran Kinesis.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.CreateStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDataStream {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <streamName>

            Where:
                streamName - The Amazon Kinesis data stream (for example,
                StockTradeStream).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
```

```
        .region(region)
        .build();
    createStream(kinesisClient, streamName);
    System.out.println("Done");
    kinesisClient.close();
}

public static void createStream(KinesisClient kinesisClient, String streamName)
{
    try {
        CreateStreamRequest streamReq = CreateStreamRequest.builder()
            .streamName(streamName)
            .shardCount(1)
            .build();

        kinesisClient.createStream(streamReq);

    } catch (KinesisException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [CreateStream](#) di Referensi AWS SDK for Java 2.x API.

## Hapus aliran

Contoh kode berikut menunjukkan cara menghapus aliran Kinesis.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DeleteStreamRequest;
```



```
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataStream {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <streamName>

            Where:
                streamName - The Amazon Kinesis data stream (for example,
StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
            .region(region)
            .build();

        deleteStream(kinesisClient, streamName);
        kinesisClient.close();
        System.out.println("Done");
    }

    public static void deleteStream(KinesisClient kinesisClient, String streamName)
    {
        try {
            DeleteStreamRequest delStream = DeleteStreamRequest.builder()
                .streamName(streamName)

```

```

        .build();

        kinesisClient.deleteStream(delStream);

    } catch (KinesisException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [DeleteStream](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan data dalam batch dari aliran

Contoh kode berikut menunjukkan cara mendapatkan data dalam batch dari aliran Kinesis.

SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.Shard;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorRequest;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorResponse;
import software.amazon.awssdk.services.kinesis.model.Record;
import software.amazon.awssdk.services.kinesis.model.GetRecordsRequest;
import software.amazon.awssdk.services.kinesis.model.GetRecordsResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development

```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class GetRecords {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <streamName>

                Where:
                streamName - The Amazon Kinesis data stream to read from (for
example, StockTradeStream).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        getStockTrades(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void getStockTrades(KinesisClient kinesisClient, String
streamName) {
        String shardIterator;
        String lastShardId = null;
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
                .streamName(streamName)
                .build();

        List<Shard> shards = new ArrayList<>();
        DescribeStreamResponse streamRes;
```

```
do {
    streamRes = kinesisClient.describeStream(describeStreamRequest);
    shards.addAll(streamRes.streamDescription().shards());

    if (shards.size() > 0) {
        lastShardId = shards.get(shards.size() - 1).shardId();
    }
} while (streamRes.streamDescription().hasMoreShards());

GetShardIteratorRequest itReq = GetShardIteratorRequest.builder()
    .streamName(streamName)
    .shardIteratorType("TRIM_HORIZON")
    .shardId(lastShardId)
    .build();

GetShardIteratorResponse shardIteratorResult =
kinesisClient.getShardIterator(itReq);
shardIterator = shardIteratorResult.shardIterator();

// Continuously read data records from shard.
List<Record> records;

// Create new GetRecordsRequest with existing shardIterator.
// Set maximum records to return to 1000.
GetRecordsRequest recordsRequest = GetRecordsRequest.builder()
    .shardIterator(shardIterator)
    .limit(1000)
    .build();

GetRecordsResponse result = kinesisClient.getRecords(recordsRequest);

// Put result into record list. Result may be empty.
records = result.records();


// Print records
for (Record record : records) {
    SdkBytes byteBuffer = record.data();
    System.out.printf("Seq No: %s - %s%n", record.sequenceNumber(), new
String(byteBuffer.asByteArray()));
}
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [GetRecords](#)
  - [GetShardIterator](#)

Masukkan data ke dalam aliran

Contoh kode berikut menunjukkan bagaimana menempatkan data ke dalam aliran Kinesis.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StockTradesWriter {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <streamName>

                Where:
```

```
        streamName - The Amazon Kinesis data stream to which records are
written (for example, StockTradeStream)
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String streamName = args[0];
    Region region = Region.US_EAST_1;
    KinesisClient kinesisClient = KinesisClient.builder()
        .region(region)
        .build();

    // Ensure that the Kinesis Stream is valid.
    validateStream(kinesisClient, streamName);
    setStockData(kinesisClient, streamName);
    kinesisClient.close();
}

public static void setStockData(KinesisClient kinesisClient, String streamName)
{
    try {
        // Repeatedly send stock trades with a 100 milliseconds wait in between.
        StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

        // Put in 50 Records for this example.
        int index = 50;
        for (int x = 0; x < index; x++) {
            StockTrade trade = stockTradeGenerator.getRandomTrade();
            sendStockTrade(trade, kinesisClient, streamName);
            Thread.sleep(100);
        }

    } catch (KinesisException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done");
}

private static void sendStockTrade(StockTrade trade, KinesisClient
kinesisClient,
```

```
        String streamName) {
        byte[] bytes = trade.toJsonAsBytes();

        // The bytes could be null if there is an issue with the JSON serialization
by
        // the Jackson JSON library.
        if (bytes == null) {
            System.out.println("Could not get JSON bytes for stock trade");
            return;
        }

        System.out.println("Putting trade: " + trade);
        PutRecordRequest request = PutRecordRequest.builder()
            .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol
as the partition key, explained in
                                                    // the Supplemental
Information section below.
            .streamName(streamName)
            .data(SdkBytes.fromByteArray(bytes))
            .build();

        try {
            kinesisClient.putRecord(request);
        } catch (KinesisException e) {
            System.err.println(e.getMessage());
        }
    }

    private static void validateStream(KinesisClient kinesisClient, String
streamName) {
        try {
            DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
                .streamName(streamName)
                .build();

            DescribeStreamResponse describeStreamResponse =
kinesisClient.describeStream(describeStreamRequest);

            if (!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
            {
                System.err.println("Stream " + streamName + " is not active. Please
wait a few moments and try again.");
            }
        }
    }
}
```

```
        System.exit(1);
    }

    } catch (KinesisException e) {
        System.err.println("Error found while describing the stream " +
streamName);
        System.err.println(e);
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [PutRecord](#) di Referensi AWS SDK for Java 2.x API.

## Contoh nirserver

### Memanggil fungsi Lambda dari pemicu Kinesis

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menerima peristiwa yang dipicu oleh menerima catatan dari aliran Kinesis. Fungsi mengambil payload Kinesis, mendekode dari Base64, dan mencatat konten rekaman.

### SDK for Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

### Mengkonsumsi acara Kinesis dengan Lambda menggunakan Java.

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
```



```
@Override
public Void handleRequest(final KinesisEvent event, final Context context) {
    LambdaLogger logger = context.getLogger();
    if (event.getRecords().isEmpty()) {
        logger.log("Empty Kinesis Event received");
        return null;
    }
    for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
        try {
            logger.log("Processed Event with EventId: "+record.getEventID());
            String data = new String(record.getKinesis().getData().array());
            logger.log("Data:"+ data);
            // TODO: Do interesting work based on the new data
        }
        catch (Exception ex) {
            logger.log("An error occurred:"+ex.getMessage());
            throw ex;
        }
    }
    logger.log("Successfully processed:"+event.getRecords().size()+" records");
    return null;
}
}
```

## Melaporkan kegagalan item batch untuk fungsi Lambda dengan pemicu Kinesis

Contoh kode berikut menunjukkan cara mengimplementasikan respons batch sebagian untuk fungsi Lambda yang menerima peristiwa dari aliran Kinesis. Fungsi melaporkan kegagalan item batch dalam respons, memberi sinyal ke Lambda untuk mencoba lagi pesan tersebut nanti.

### SDK for Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

Melaporkan kegagalan item batch Kinesis dengan Lambda menggunakan Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
ArrayList<>();
        String curRecordSequenceNumber = "";

        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
immediately.
                Lambda will immediately begin to retry processing from this
failed item onwards. */
                batchItemFailures.add(new
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

        return new StreamsEventResponse(batchItemFailures);
    }
}
```

## AWS KMS contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with AWS KMS.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

### Tindakan

Buat hibah untuk kunci

Contoh kode berikut menunjukkan cara membuat hibah untuk kunci KMS.

SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.CreateGrantRequest;
import software.amazon.awssdk.services.kms.model.CreateGrantResponse;
import software.amazon.awssdk.services.kms.model.KmsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateGrant {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <keyId> <granteePrincipal> <operation>\s

            Where:
                keyId - The unique identifier for the customer master key (CMK)
that the grant applies to.\s
                granteePrincipal - The principal that is given permission to
perform the operations that the grant permits.\s
                operation - An operation (for example, Encrypt).\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String keyId = args[0];
        String granteePrincipal = args[1];
        String operation = args[2];
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        String grantId = createGrant(kmsClient, keyId, granteePrincipal, operation);
        System.out.printf("Successfully created a grant with ID %s%n", grantId);
        kmsClient.close();
    }

    public static String createGrant(KmsClient kmsClient, String keyId, String
granteePrincipal, String operation) {
        try {
            CreateGrantRequest grantRequest = CreateGrantRequest.builder()
                .keyId(keyId)
                .granteePrincipal(granteePrincipal)
```

```

        .operationsWithStrings(operation)
        .build();

        CreateGrantResponse response = kmsClient.createGrant(grantRequest);
        return response.grantId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
}

```

- Untuk detail API, lihat [CreateGrant](#) di Referensi AWS SDK for Java 2.x API.

## Buat kunci

Contoh kode berikut menunjukkan cara membuat AWS KMS key.

## SDK for Java 2.x

### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.CreateKeyRequest;
import software.amazon.awssdk.services.kms.model.CustomerMasterKeySpec;
import software.amazon.awssdk.services.kms.model.CreateKeyResponse;
import software.amazon.awssdk.services.kms.model.KmsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:

```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateCustomerKey {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        String keyDesc = "Created by the AWS KMS API";
        System.out.println("The key id is " + createKey(kmsClient, keyDesc));
        kmsClient.close();
    }

    public static String createKey(KmsClient kmsClient, String keyDesc) {
        try {
            CreateKeyRequest keyRequest = CreateKeyRequest.builder()
                .description(keyDesc)
                .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
                .keyUsage("ENCRYPT_DECRYPT")
                .build();

            CreateKeyResponse result = kmsClient.createKey(keyRequest);
            System.out.printf("Created a customer key with id \"%s\"%n",
result.keyMetadata().arn());
            return result.keyMetadata().keyId();

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Untuk detail API, lihat [CreateKey](#) di Referensi AWS SDK for Java 2.x API.

Buat alias untuk kunci

Contoh kode berikut menunjukkan cara membuat alias untuk kunci kunci KMS.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.CreateAliasRequest;
import software.amazon.awssdk.services.kms.model.KmsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAlias {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <targetKeyId> <aliasName>\s

            Where:
                targetKeyId - The key ID or the Amazon Resource Name (ARN) of
the customer master key (CMK).\s
                aliasName - An alias name (for example, alias/myAlias).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String targetKeyId = args[0];
        String aliasName = args[1];
```

```
    Region region = Region.US_WEST_2;
    KmsClient kmsClient = KmsClient.builder()
        .region(region)
        .build();

    createCustomAlias(kmsClient, targetKeyId, aliasName);
    kmsClient.close();
}

public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,
String aliasName) {
    try {
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        kmsClient.createAlias(aliasRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [CreateAlias](#) di Referensi AWS SDK for Java 2.x API.

## Dekripsi ciphertext

Contoh kode berikut menunjukkan cara mendekripsi ciphertext yang dienkrpsi oleh kunci KMS.

## SDK for Java 2.x

### Note

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



```
public static void decryptData(KmsClient kmsClient, SdkBytes encryptedData,
String keyId) {
    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
        decryptResponse.plaintext();


    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [Mendekripsi](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan kunci

Contoh kode berikut menunjukkan bagaimana mendeskripsikan kunci KMS.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.DescribeKeyRequest;
import software.amazon.awssdk.services.kms.model.DescribeKeyResponse;
import software.amazon.awssdk.services.kms.model.KmsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DescribeKey {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <keyId>\s

            Where:
                keyId - A key id value to describe (for example,
                xxxxxbcd-12ab-34cd-56ef-1234567890ab).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String keyId = args[0];
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        describeSpecifcKey(kmsClient, keyId);
        kmsClient.close();
    }

    public static void describeSpecifcKey(KmsClient kmsClient, String keyId) {
        try {
            DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
                .keyId(keyId)
                .build();

            DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
            System.out.println("The key description is " +
response.keyMetadata().description());
            System.out.println("The key ARN is " + response.keyMetadata().arn());

        } catch (KmsException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DescribeKey](#) di Referensi AWS SDK for Java 2.x API.

## Nonaktifkan kunci

Contoh kode berikut menunjukkan cara menonaktifkan kunci KMS.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.DisableKeyRequest;
import software.amazon.awssdk.services.kms.model.KmsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DisableCustomerKey {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <keyId>\s
```

```
        Where:
            keyId - A key id value to disable (for example,
xxxxxbcd-12ab-34cd-56ef-1234567890ab).\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String keyId = args[0];
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        disableKey(kmsClient, keyId);
        kmsClient.close();
    }

    public static void disableKey(KmsClient kmsClient, String keyId) {
        try {
            DisableKeyRequest keyRequest = DisableKeyRequest.builder()
                .keyId(keyId)
                .build();

            kmsClient.disableKey(keyRequest);

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DisableKey](#) di Referensi AWS SDK for Java 2.x API.

## Aktifkan kunci

Contoh kode berikut menunjukkan cara mengaktifkan kunci KMS.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.EnableKeyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnableCustomerKey {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <keyId>\s

            Where:
                keyId - A key id value to enable (for example,
                xxxxxbcd-12ab-34cd-56ef-1234567890ab).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String keyId = args[0];
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
```

```
        .build();

        enableKey(kmsClient, keyId);
        kmsClient.close();
    }

    public static void enableKey(KmsClient kmsClient, String keyId) {
        try {
            EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
                .keyId(keyId)
                .build();

            kmsClient.enableKey(enableKeyRequest);

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [EnableKey](#) di Referensi AWS SDK for Java 2.x API.

Enkripsi teks menggunakan kunci

Contoh kode berikut menunjukkan cara mengenkripsi teks menggunakan kunci KMS.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.EncryptRequest;
import software.amazon.awssdk.services.kms.model.EncryptResponse;
import software.amazon.awssdk.services.kms.model.KmsException;
```

```
import software.amazon.awssdk.services.kms.model.DecryptRequest;
import software.amazon.awssdk.services.kms.model.DecryptResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EncryptDataKey {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <keyId>\s

                Where:
                keyId - A key id value to use to encrypt/decrypt the data (for
                example, xxxxxbcd-12ab-34cd-56ef-1234567890ab).\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String keyId = args[0];
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        SdkBytes encryData = encryptData(kmsClient, keyId);
        decryptData(kmsClient, encryData, keyId);
        System.out.println("Done");
        kmsClient.close();
    }

    public static SdkBytes encryptData(KmsClient kmsClient, String keyId) {
        try {
```

```
        SdkBytes myBytes = SdkBytes.fromByteArray(new byte[] { 1, 2, 3, 4, 5, 6,
7, 8, 9, 0 });
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        EncryptResponse response = kmsClient.encrypt(encryptRequest);
        String algorithm = response.encryptionAlgorithm().toString();
        System.out.println("The encryption algorithm is " + algorithm);

        // Get the encrypted data.
        SdkBytes encryptedData = response.ciphertextBlob();
        return encryptedData;

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return null;
}

public static void decryptData(KmsClient kmsClient, SdkBytes encryptedData,
String keyId) {
    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
        decryptResponse.plaintext();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [Enkripsi](#) di Referensi AWS SDK for Java 2.x API.



## Daftar alias untuk kunci

Contoh kode berikut menunjukkan bagaimana untuk daftar alias untuk kunci KMS.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.AliasListEntry;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.ListAliasesRequest;
import software.amazon.awssdk.services.kms.model.ListAliasesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListAliases {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        listAllAliases(kmsClient);
        kmsClient.close();
    }

    public static void listAllAliases(KmsClient kmsClient) {
        try {
            ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
                .limit(15)

```

```

        .build();

        ListAliasesResponse aliasesResponse =
kmsClient.listAliases(aliasesRequest);
        List<AliasListEntry> aliases = aliasesResponse.aliases();
        for (AliasListEntry alias : aliases) {
            System.out.println("The alias name is: " + alias.aliasName());
        }

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}

```

- Untuk detail API, lihat [ListAliases](#) di Referensi AWS SDK for Java 2.x API.

## Daftar hibah untuk kunci

Contoh kode berikut menunjukkan bagaimana daftar hibah untuk kunci KMS.

### SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.GrantListEntry;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.ListGrantsRequest;
import software.amazon.awssdk.services.kms.model.ListGrantsResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.

```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListGrants {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <keyId>\s

            Where:
                keyId - a key id value to use (for example,
                xxxxxbcd-12ab-34cd-56ef-1234567890ab).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String keyId = args[0];
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        displayGrantIds(kmsClient, keyId);
        kmsClient.close();
    }

    public static void displayGrantIds(KmsClient kmsClient, String keyId) {
        try {
            ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
                .keyId(keyId)
                .limit(15)
                .build();

            ListGrantsResponse response = kmsClient.listGrants(grantsRequest);
            List<GrantListEntry> grants = response.grants();
            for (GrantListEntry grant : grants) {
                System.out.println("The grant Id is : " + grant.grantId());
            }
        }
    }
}
```

```
        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListGrants](#) di Referensi AWS SDK for Java 2.x API.

## Daftar kunci

Contoh kode berikut menunjukkan bagaimana untuk daftar kunci KMS.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.KeyListEntry;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.model.ListKeysResponse;
import software.amazon.awssdk.services.kms.model.KmsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListKeys {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
```

```
KmsClient kmsClient = KmsClient.builder()
    .region(region)
    .build();

listAllKeys(kmsClient);
kmsClient.close();
}

public static void listAllKeys(KmsClient kmsClient) {
    try {
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        ListKeysResponse keysResponse = kmsClient.listKeys(listKeysRequest);
        List<KeyListEntry> keyListEntries = keysResponse.keys();
        for (KeyListEntry key : keyListEntries) {
            System.out.println("The key ARN is: " + key.keyArn());
            System.out.println("The key Id is: " + key.keyId());
        }

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ListKeys](#) di Referensi AWS SDK for Java 2.x API.

## Contoh Lambda menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan Lambda AWS SDK for Java 2.x with.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

## Memulai

### Halo Lambda

Contoh kode berikut menunjukkan cara memulai menggunakan Lambda.

### SDK for Java 2.x

#### Note

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

```
package com.example.lambda;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.ListFunctionsResponse;
import software.amazon.awssdk.services.lambda.model.FunctionConfiguration;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListLambdaFunctions {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        listFunctions(awsLambda);
        awsLambda.close();
    }
}
```

```
    }

    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config : list) {
                System.out.println("The function name is " + config.functionName());
            }
        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListFunctions](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

Buat fungsi

Contoh kode berikut menunjukkan cara membuat fungsi Lambda.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.core.SdkBytes;
```

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.CreateFunctionRequest;
import software.amazon.awssdk.services.lambda.model.FunctionCode;
import software.amazon.awssdk.services.lambda.model.CreateFunctionResponse;
import software.amazon.awssdk.services.lambda.model.GetFunctionRequest;
import software.amazon.awssdk.services.lambda.model.GetFunctionResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.Runtime;
import software.amazon.awssdk.services.lambda.waiters.LambdaWaiter;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;

/**
 * This code example requires a ZIP or JAR that represents the code of the
 * Lambda function.
 * If you do not have a ZIP or JAR, please refer to the following document:
 *
 * https://github.com/aws-doc-sdk-examples/tree/master/javav2/usecases/creating\_workflows\_stepfunctions
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateFunction {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <functionName> <filePath> <role> <handler>\s

            Where:
                functionName - The name of the Lambda function.\s
                filePath - The path to the ZIP or JAR where the code is located.
\s

                role - The role ARN that has Lambda permissions.\s
```



```
        handler - The fully qualified method name (for example,
example.Handler::handleRequest). \s
        """;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String functionName = args[0];
    String filePath = args[1];
    String role = args[2];
    String handler = args[3];
    Region region = Region.US_WEST_2;
    LambdaClient awsLambda = LambdaClient.builder()
        .region(region)
        .build();

    createLambdaFunction(awsLambda, functionName, filePath, role, handler);
    awsLambda.close();
}

public static void createLambdaFunction(LambdaClient awsLambda,
    String functionName,
    String filePath,
    String role,
    String handler) {

    try {
        LambdaWaiter waiter = awsLambda.waiter();
        InputStream is = new FileInputStream(filePath);
        SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

        FunctionCode code = FunctionCode.builder()
            .zipFile(fileToUpload)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .functionName(functionName)
            .description("Created by the Lambda Java API")
            .code(code)
            .handler(handler)
            .runtime(Runtime.JAVA8)
            .role(role)
    }
```

```

        .build();

        // Create a Lambda function using a waiter.
        CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
        GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();
        WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The function ARN is " +
functionResponse.functionArn());

    } catch (LambdaException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [CreateFunction](#) di Referensi AWS SDK for Java 2.x API.

## Hapus fungsi

Contoh kode berikut menunjukkan cara menghapus fungsi Lambda.

### SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.DeleteFunctionRequest;
import software.amazon.awssdk.services.lambda.model.LambdaException;

/**

```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DeleteFunction {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <functionName>\s

            Where:
                functionName - The name of the Lambda function.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        Region region = Region.US_EAST_1;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        deleteLambdaFunction(awsLambda, functionName);
        awsLambda.close();
    }

    public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                .functionName(functionName)
                .build();

            awsLambda.deleteFunction(request);
            System.out.println("The " + functionName + " function was deleted");
        } catch (LambdaException e) {
```

```

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}

```

- Untuk detail API, lihat [DeleteFunction](#) di Referensi AWS SDK for Java 2.x API.

## Memanggil fungsi

Contoh kode berikut menunjukkan cara memanggil fungsi Lambda.

### SDK for Java 2.x

#### Note

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

```

import org.json.JSONObject;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.InvokeRequest;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lambda.model.InvokeResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;

public class LambdaInvoke {

    /**
     * Function names appear as
     * arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
     * you can retrieve the value by looking at the function in the AWS Console
     *
     * Also, set up your development environment, including your credentials.
     *
     * For information, see this documentation topic:
     *
     * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.
     */
}

```

```
* html
*/

public static void main(String[] args) {
    final String usage = ""

        Usage:
            <functionName>\s

        Where:
            functionName - The name of the Lambda function\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String functionName = args[0];
    Region region = Region.US_WEST_2;
    LambdaClient awsLambda = LambdaClient.builder()
        .region(region)
        .build();

    invokeFunction(awsLambda, functionName);
    awsLambda.close();
}

public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res = null;
    try {
        // Need a SdkBytes instance for the payload.
        JSONObject jsonObj = new JSONObject();
        jsonObj.put("inputValue", "2000");
        String json = jsonObj.toString();
        SdkBytes payload = SdkBytes.fromUtf8String(json);

        // Setup an InvokeRequest.
        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
            .build();
```

```
        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String();
        System.out.println(value);

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [Memanggil di Referensi AWS SDK for Java 2.x API](#).

## Skenario

Memulai dengan fungsi

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- 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 for Java 2.x

### Note

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

```
/*
 * Lambda function names appear as:
```

```

*
*  arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
*
*  To find this value, look at the function in the AWS Management Console.
*
*  Before running this Java code example, set up your development environment,
including your credentials.
*
*  For more information, see this documentation topic:
*
*  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
*  This example performs the following tasks:
*
*  1. Creates an AWS Lambda function.
*  2. Gets a specific AWS Lambda function.
*  3. Lists all Lambda functions.
*  4. Invokes a Lambda function.
*  5. Updates the Lambda function code and invokes it again.
*  6. Updates a Lambda function's configuration value.
*  7. Deletes a Lambda function.
*/

```

```

public class LambdaScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <functionName> <filePath> <role> <handler> <bucketName> <key>\s

            Where:
                functionName - The name of the Lambda function.\s
                filePath - The path to the .zip or .jar where the code is
located.\s
                role - The AWS Identity and Access Management (IAM) service role
that has Lambda permissions.\s
                handler - The fully qualified method name (for example,
example.Handler::handleRequest).\s
                bucketName - The Amazon Simple Storage Service (Amazon S3)
bucket name that contains the .zip or .jar used to update the Lambda function's
code.\s

```

```
        key - The Amazon S3 key name that represents the .zip or .jar
(for example, LambdaHello-1.0-SNAPSHOT.jar).
        """;

    if (args.length != 6) {
        System.out.println(usage);
        System.exit(1);
    }

    String functionName = args[0];
    String filePath = args[1];
    String role = args[2];
    String handler = args[3];
    String bucketName = args[4];
    String key = args[5];

    Region region = Region.US_WEST_2;
    LambdaClient awsLambda = LambdaClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the AWS Lambda example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Create an AWS Lambda function.");
    String funArn = createLambdaFunction(awsLambda, functionName, filePath,
role, handler);
    System.out.println("The AWS Lambda ARN is " + funArn);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Get the " + functionName + " AWS Lambda function.");
    getFunction(awsLambda, functionName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. List all AWS Lambda functions.");
    listFunctions(awsLambda);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Invoke the Lambda function.");
```



```
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update the Lambda function code and invoke it
again.");
        updateFunctionCode(awsLambda, functionName, bucketName, key);
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Update a Lambda function's configuration value.");
        updateFunctionConfiguration(awsLambda, functionName, handler);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the AWS Lambda function.");
        LambdaScenario.deleteLambdaFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Lambda scenario completed successfully");
        System.out.println(DASHES);
        awsLambda.close();
    }

    public static String createLambdaFunction(LambdaClient awsLambda,
        String functionName,
        String filePath,
        String role,
        String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

            FunctionCode code = FunctionCode.builder()
                .zipFile(fileToUpload)
                .build();
```

```
        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .functionName(functionName)
            .description("Created by the Lambda Java API")
            .code(code)
            .handler(handler)
            .runtime(Runtime.JAVA8)
            .role(role)
            .build();

        // Create a Lambda function using a waiter
        CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
        GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();
        WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        return functionResponse.functionArn();

    } catch (LambdaException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void getFunction(LambdaClient awsLambda, String functionName) {
    try {
        GetFunctionRequest functionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();

        GetFunctionResponse response = awsLambda.getFunction(functionRequest);
        System.out.println("The runtime of this Lambda function is " +
response.configuration().runtime());

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
public static void listFunctions(LambdaClient awsLambda) {
    try {
        ListFunctionsResponse functionResult = awsLambda.listFunctions();
        List<FunctionConfiguration> list = functionResult.functions();
        for (FunctionConfiguration config : list) {
            System.out.println("The function name is " + config.functionName());
        }
    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res;
    try {
        // Need a SdkBytes instance for the payload.
        JSONObject jsonObj = new JSONObject();
        jsonObj.put("inputValue", "2000");
        String json = jsonObj.toString();
        SdkBytes payload = SdkBytes.fromUtf8String(json);

        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
            .build();

        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String();
        System.out.println(value);
    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateFunctionCode(LambdaClient awsLambda, String
functionName, String bucketName, String key) {
    try {
        LambdaWaiter waiter = awsLambda.waiter();
```

```
        UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
            .functionName(functionName)
            .publish(true)
            .s3Bucket(bucketName)
            .s3Key(key)
            .build();

        UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest);
        GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
            .functionName(functionName)
            .build();

        WaiterResponse<GetFunctionConfigurationResponse> waiterResponse = waiter
            .waitUntilFunctionUpdated(getFunctionConfigRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The last modified value is " +
response.lastModified());

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
    try {
        UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
            .functionName(functionName)
            .handler(handler)
            .runtime(Runtime.JAVA11)
            .build();

        awsLambda.updateFunctionConfiguration(configurationRequest);

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("The " + functionName + " function was deleted");

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Memohon](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

## Contoh nirserver

### Memanggil fungsi Lambda dari pemicu Kinesis

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menerima peristiwa yang dipicu oleh menerima catatan dari aliran Kinesis. Fungsi mengambil payload Kinesis, mendekode dari Base64, dan mencatat konten rekaman.

## SDK for Java 2.x

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

## Mengkonsumsi acara Kinesis dengan Lambda menggunakan Java.

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+" records");
        return null;
    }
}
```

## Menginvokasi fungsi Lambda dari pemicu Amazon S3

Contoh kode berikut menunjukkan cara mengimplementasikan fungsi Lambda yang menerima peristiwa yang dipicu dengan mengunggah objek ke bucket S3. Fungsi ini mengambil nama bucket S3 dan kunci objek dari parameter peristiwa dan memanggil Amazon S3 API untuk mengambil dan mencatat jenis konten objek.

### SDK for Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

## Menggunakan peristiwa S3 dengan Lambda menggunakan Java.

```
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
    com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotificat

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
            String srcBucket = record.getS3().getBucket().getName();
```

```
String srcKey = record.getS3().getObject().getUrlDecodedKey();

S3Client s3Client = S3Client.builder().build();
HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());

return "Ok";
} catch (Exception e) {
throw new RuntimeException(e);
}
}

private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
.bucket(bucket)
.key(key)
.build();
return s3Client.headObject(headObjectRequest);
}
}
```

## Memanggil fungsi Lambda dari pemicu Amazon SNS

Contoh kode berikut menunjukkan cara menerapkan fungsi Lambda yang menerima peristiwa yang dipicu dengan menerima pesan dari topik SNS. Fungsi mengambil pesan dari parameter peristiwa dan mencatat konten setiap pesan.

### SDK for Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

Mengonsumsi acara SNS dengan Lambda menggunakan Java.



```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;

import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```

## Memanggil fungsi Lambda dari pemicu Amazon SQS

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menerima peristiwa yang dipicu oleh menerima pesan dari antrian SQS. Fungsi mengambil pesan dari parameter peristiwa dan mencatat konten setiap pesan.

SDK for Java 2.x

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

Mengonsumsi acara SQS dengan Lambda menggunakan Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }
    }
}
```

```
}  
}
```

## Melaporkan kegagalan item batch untuk fungsi Lambda dengan pemicu Kinesis

Contoh kode berikut menunjukkan cara mengimplementasikan respons batch sebagian untuk fungsi Lambda yang menerima peristiwa dari aliran Kinesis. Fungsi melaporkan kegagalan item batch dalam respons, memberi sinyal ke Lambda untuk mencoba lagi pesan tersebut nanti.

### SDK for Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

## Melaporkan kegagalan item batch Kinesis dengan Lambda menggunakan Java.

```
import com.amazonaws.services.lambda.runtime.Context;  
import com.amazonaws.services.lambda.runtime.RequestHandler;  
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;  
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;  
  
import java.io.Serializable;  
import java.util.ArrayList;  
import java.util.List;  
  
public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,  
StreamsEventResponse> {  
  
    @Override  
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {  
  
        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new  
ArrayList<>();  
        String curRecordSequenceNumber = "";  
  
        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :  
input.getRecords()) {  
            try {
```

```
        //Process your record
        KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
        curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

    } catch (Exception e) {
        /* Since we are working with streams, we can return the failed item
        immediately.
           Lambda will immediately begin to retry processing from this
        failed item onwards. */
        batchItemFailures.add(new
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
        return new StreamsEventResponse(batchItemFailures);
    }
}

return new StreamsEventResponse(batchItemFailures);
}
}
```

## Melaporkan kegagalan item batch untuk fungsi Lambda dengan pemicu Amazon SQS

Contoh kode berikut menunjukkan cara mengimplementasikan respons batch sebagian untuk fungsi Lambda yang menerima peristiwa dari antrian SQS. Fungsi melaporkan kegagalan item batch dalam respons, memberi sinyal ke Lambda untuk mencoba lagi pesan tersebut nanti.

### SDK for Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

## Melaporkan kegagalan item batch SQS dengan Lambda menggunakan Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;
```

```
import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {

        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
ArrayList<SQSBatchResponse.BatchItemFailure>();
        String messageId = "";
        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
                messageId = message.getMessageId();
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
SQSBatchResponse.BatchItemFailure(messageId));
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }
}
```

## MediaConvert contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with MediaConvert.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

### Topik

- [Tindakan](#)

## Tindakan

Buat pekerjaan transcoding

Contoh kode berikut menunjukkan cara membuat pekerjaan AWS Elemental MediaConvert transcoding.

SDK for Java 2.x

### Note

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

```
package com.example.mediaconvert;

import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.Output;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroup;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.HlsGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupType;
import software.amazon.awssdk.services.mediaconvert.model.HlsDirectoryStructure;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestDurationFormat;
import software.amazon.awssdk.services.mediaconvert.model.HlsStreamInfResolution;
import software.amazon.awssdk.services.mediaconvert.model.HlsClientCache;
import software.amazon.awssdk.services.mediaconvert.model.HlsCaptionLanguageSetting;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestCompression;
import software.amazon.awssdk.services.mediaconvert.model.HlsCodecSpecification;
import software.amazon.awssdk.services.mediaconvert.model.HlsOutputSelection;
import software.amazon.awssdk.services.mediaconvert.model.HlsProgramDateTime;
import software.amazon.awssdk.services.mediaconvert.model.HlsTimedMetadataId3Frame;
```

```
import software.amazon.awssdk.services.mediaconvert.model.HlsSegmentControl;
import software.amazon.awssdk.services.mediaconvert.model.FileGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.ContainerSettings;
import software.amazon.awssdk.services.mediaconvert.model.VideoDescription;
import software.amazon.awssdk.services.mediaconvert.model.ContainerType;
import software.amazon.awssdk.services.mediaconvert.model.ScalingBehavior;
import software.amazon.awssdk.services.mediaconvert.model.VideoTimecodeInsertion;
import software.amazon.awssdk.services.mediaconvert.model.ColorMetadata;
import software.amazon.awssdk.services.mediaconvert.model.RespondToAfd;
import software.amazon.awssdk.services.mediaconvert.model.AfdSignaling;
import software.amazon.awssdk.services.mediaconvert.model.DropFrameTimecode;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264Settings;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodec;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.H264RateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.H264QualityTuningLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264SceneChangeDetect;
import
    software.amazon.awssdk.services.mediaconvert.model.AacAudioDescriptionBroadcasterMix;
import software.amazon.awssdk.services.mediaconvert.model.H264ParControl;
import software.amazon.awssdk.services.mediaconvert.model.AacRawFormat;
import software.amazon.awssdk.services.mediaconvert.model.H264QvbrSettings;
import
    software.amazon.awssdk.services.mediaconvert.model.H264FramerateConversionAlgorithm;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264FramerateControl;
import software.amazon.awssdk.services.mediaconvert.model.AacCodingMode;
import software.amazon.awssdk.services.mediaconvert.model.H264Telecine;
import
    software.amazon.awssdk.services.mediaconvert.model.H264FlickerAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264GopSizeUnits;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.H264GopBReference;
import software.amazon.awssdk.services.mediaconvert.model.AudioTypeControl;
import software.amazon.awssdk.services.mediaconvert.model.AntiAlias;
import software.amazon.awssdk.services.mediaconvert.model.H264SlowPal;
import
    software.amazon.awssdk.services.mediaconvert.model.H264SpatialAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264Syntax;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Settings;
import software.amazon.awssdk.services.mediaconvert.model.InputDenoiseFilter;
import
    software.amazon.awssdk.services.mediaconvert.model.H264TemporalAdaptiveQuantization;
```

```
import software.amazon.awssdk.services.mediaconvert.model.CreateJobResponse;
import
    software.amazon.awssdk.services.mediaconvert.model.H264UnregisteredSeiTimecode;
import software.amazon.awssdk.services.mediaconvert.model.H264EntropyEncoding;
import software.amazon.awssdk.services.mediaconvert.model.InputPsiControl;
import software.amazon.awssdk.services.mediaconvert.model.ColorSpace;
import software.amazon.awssdk.services.mediaconvert.model.H264RepeatPps;
import software.amazon.awssdk.services.mediaconvert.model.H264FieldEncoding;
import software.amazon.awssdk.services.mediaconvert.model.M3u8NielsenId3;
import software.amazon.awssdk.services.mediaconvert.model.InputDeblockFilter;
import software.amazon.awssdk.services.mediaconvert.model.InputRotate;
import software.amazon.awssdk.services.mediaconvert.model.H264DynamicSubGop;
import software.amazon.awssdk.services.mediaconvert.model.TimedMetadata;
import software.amazon.awssdk.services.mediaconvert.model.JobSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioDefaultSelection;
import software.amazon.awssdk.services.mediaconvert.model.VideoSelector;
import software.amazon.awssdk.services.mediaconvert.model.AacSpecification;
import software.amazon.awssdk.services.mediaconvert.model.Input;
import software.amazon.awssdk.services.mediaconvert.model.OutputSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264AdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.AudioLanguageCodeControl;
import software.amazon.awssdk.services.mediaconvert.model.InputFilterEnable;
import software.amazon.awssdk.services.mediaconvert.model.AudioDescription;
import software.amazon.awssdk.services.mediaconvert.model.H264InterlaceMode;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.AacSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodec;
import software.amazon.awssdk.services.mediaconvert.model.AacRateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.AacCodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.HlsIFrameOnlyManifest;
import software.amazon.awssdk.services.mediaconvert.model.FrameCaptureSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioSelector;
import software.amazon.awssdk.services.mediaconvert.model.M3u8PcrControl;
import software.amazon.awssdk.services.mediaconvert.model.InputTimecodeSource;
import software.amazon.awssdk.services.mediaconvert.model.HlsSettings;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Scte35Source;

/**
 * Create a MediaConvert job. Must supply MediaConvert access role Amazon
 * Resource Name (ARN), and a
 * valid video input file via Amazon S3 URL.
 *
 * Also, set up your development environment, including your credentials.
 */
```



```

* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
*/
public class CreateJob {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <mcRoleARN> <fileInput>\s

            Where:
                mcRoleARN - The MediaConvert Role ARN.\s
                fileInput - The URL of an Amazon S3 bucket
where the input file is located.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String mcRoleARN = args[0];
        String fileInput = args[1];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
            .region(region)
            .build();

        String id = createMediaJob(mc, mcRoleARN, fileInput);
        System.out.println("MediaConvert job created. Job Id = " + id);
        mc.close();
    }

    public static String createMediaJob(MediaConvertClient mc, String mcRoleARN,
String fileInput) {

        String s3path = fileInput.substring(0, fileInput.lastIndexOf('/') +
1) + "javasdk/out/";
        String fileOutput = s3path + "index";
        String thumbsOutput = s3path + "thumbs/";
        String mp4Output = s3path + "mp4/";

```

```

        try {
            DescribeEndpointsResponse res = mc

.describeEndpoints(DescribeEndpointsRequest.builder().maxResults(20).build());

            if (res.endpoints().size() <= 0) {
                System.out.println("Cannot find MediaConvert service
endpoint URL!");

                System.exit(1);
            }
            String endpointURL = res.endpoints().get(0).url();
            System.out.println("MediaConvert service URL: " +
endpointURL);

            System.out.println("MediaConvert role arn: " + mcRoleARN);
            System.out.println("MediaConvert input file: " + fileInput);
            System.out.println("MediaConvert output path: " + s3path);

            MediaConvertClient emc = MediaConvertClient.builder()
                .region(Region.US_WEST_2)
                .endpointOverride(URI.create(endpointURL))
                .build();

            // output group Preset HLS low profile
            Output hlsLow = createOutput("hls_low", "_low", "_$dt$",
750000, 7, 1920, 1080, 640);
            // output group Preset HLS media profile
            Output hlsMedium = createOutput("hls_medium", "_medium", "_
$dt$", 1200000, 7, 1920, 1080, 1280);
            // output group Preset HLS high profole
            Output hlsHigh = createOutput("hls_high", "_high", "_$dt$",
3500000, 8, 1920, 1080, 1920);

            OutputGroup appleHLS = OutputGroup.builder().name("Apple
HLS").customName("Example")

            .outputGroupSettings(OutputGroupSettings.builder()

            .type(OutputGroupType.HLS_GROUP_SETTINGS)

            .hlsGroupSettings(HlsGroupSettings.builder()

            .directoryStructure(

                HlsDirectoryStructure.SINGLE_DIRECTORY)

```

```
.manifestDurationFormat(  
    HlsManifestDurationFormat.INTEGER)  
.streamInfResolution(  
    HlsStreamInfResolution.INCLUDE)  
.clientCache(HlsClientCache.ENABLED)  
.captionLanguageSetting(  
    HlsCaptionLanguageSetting.OMIT)  
.manifestCompression(  
    HlsManifestCompression.NONE)  
.codecSpecification(  
    HlsCodecSpecification.RFC_4281)  
.outputSelection(  
    HlsOutputSelection.MANIFESTS_AND_SEGMENTS)  
.programDateTime(HlsProgramDateTime.EXCLUDE)  
.programDateTimePeriod(600)  
.timedMetadataId3Frame(  
    HlsTimedMetadataId3Frame.PRIV)  
.timedMetadataId3Period(10)  
.destination(fileOutput)  
.segmentControl(HlsSegmentControl.SEGMENTED_FILES)  
.minFinalSegmentLength((double) 0)  
.segmentLength(4).minSegmentLength(0).build())
```

```
                .build())
                .outputs(hlsLow, hlsMedium,
hlsHigh).build());

        OutputGroup fileMp4 = OutputGroup.builder().name("File
Group").customName("mp4")

        .outputGroupSettings(OutputGroupSettings.builder()

        .type(OutputGroupType.FILE_GROUP_SETTINGS)

        .fileGroupSettings(FileGroupSettings.builder()

        .destination(mp4Output).build())

                .build())
                .outputs(Output.builder().extension("mp4")

        .containerSettings(ContainerSettings.builder()

        .container(ContainerType.MP4).build())

        .videoDescription(VideoDescription.builder().width(1280)

                .height(720)

        .scalingBehavior(ScalingBehavior.DEFAULT)

        .sharpness(50).antiAlias(AntiAlias.ENABLED)

        .timecodeInsertion(

                VideoTimecodeInsertion.DISABLED)

        .colorMetadata(ColorMetadata.INSERT)

        .respondToAfd(RespondToAfd.NONE)

        .afdSignaling(AfdSignaling.NONE)

        .dropFrameTimecode(DropFrameTimecode.ENABLED)

        .codecSettings(VideoCodecSettings.builder()

                .codec(VideoCodec.H_264)
```

```
.h264Settings(H264Settings
    .builder()
    .rateControlMode(
        H264RateControlMode.QVBR)
    .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
    .qualityTuningLevel(
        H264QualityTuningLevel.SINGLE_PASS)
    .qvbrSettings(
        H264QvbrSettings.builder()
            .qvbrQualityLevel(
                8)
            .build())
    .codecLevel(H264CodecLevel.AUTO)
    .codecProfile(H264CodecProfile.MAIN)
    .maxBitrate(2400000)
    .framerateControl(
        H264FramerateControl.INITIALIZE_FROM_SOURCE)
    .gopSize(2.0)
    .gopSizeUnits(H264GopSizeUnits.SECONDS)
    .numberBFramesBetweenReferenceFrames(
        2)
    .gopClosedCadence(
```

```
1)

.gopBReference(H264GopBReference.DISABLED)

.slowPal(H264SlowPal.DISABLED)

.syntax(H264Syntax.DEFAULT)

.numberReferenceFrames(

3)

.dynamicSubGop(H264DynamicSubGop.STATIC)

.fieldEncoding(H264FieldEncoding.PAFF)

.sceneChangeDetect(

    H264SceneChangeDetect.ENABLED)

.minIInterval(0)

.telecine(H264Telecine.NONE)

.framerateConversionAlgorithm(

    H264FramerateConversionAlgorithm.DUPLICATE_DROP)

.entropyEncoding(

    H264EntropyEncoding.CABAC)

.slices(1)

.unregisterSeiTimecode(

    H264UnregisteredSeiTimecode.DISABLED)

.repeatPps(H264RepeatPps.DISABLED)

.adaptiveQuantization(

    H264AdaptiveQuantization.HIGH)
```

```
        .spatialAdaptiveQuantization(
            H264SpatialAdaptiveQuantization.ENABLED)
        .temporalAdaptiveQuantization(
            H264TemporalAdaptiveQuantization.ENABLED)
        .flickerAdaptiveQuantization(
            H264FlickerAdaptiveQuantization.DISABLED)
        .softness(0)
        .interlaceMode(H264InterlaceMode.PROGRESSIVE)
        .build()

    .build()
    .build()

    .audioDescriptions(AudioDescription.builder()
        .audioTypeControl(AudioTypeControl.FOLLOW_INPUT)
        .languageCodeControl(
            AudioLanguageCodeControl.FOLLOW_INPUT)
        .codecSettings(AudioCodecSettings.builder()
            .codec(AudioCodec.AAC)
            .aacSettings(AacSettings
                .builder()
                    .codecProfile(AacCodecProfile.LC)
                    .rateControlMode(
                        AacRateControlMode.CBR)
```

```

        .codingMode(AacCodingMode.CODING_MODE_2_0)

        .sampleRate(44100)

        .bitrate(160000)

        .rawFormat(AacRawFormat.NONE)

        .specification(AacSpecification.MPEG4)

        .audioDescriptionBroadcasterMix(

            AacAudioDescriptionBroadcasterMix.NORMAL)

        .build())

    .build()

                                                    .build()

                                                    .build()

                .build();
        OutputGroup thumbs = OutputGroup.builder().name("File
Group").customName("thumbs")

        .outputGroupSettings(OutputGroupSettings.builder()

        .type(OutputGroupType.FILE_GROUP_SETTINGS)

        .fileGroupSettings(FileGroupSettings.builder()

        .destination(thumbsOutput).build())

                                                    .build())

                .outputs(Output.builder().extension("jpg")

        .containerSettings(ContainerSettings.builder()

        .container(ContainerType.RAW).build())

        .videoDescription(VideoDescription.builder()

        .scalingBehavior(ScalingBehavior.DEFAULT)

        .sharpness(50).antiAlias(AntiAlias.ENABLED)

```



```

.timecodeInsertion(
    VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

    .codec(VideoCodec.FRAME_CAPTURE)

    .frameCaptureSettings(
        FrameCaptureSettings
            .builder()

            .framerateNumerator(
                1)

            .framerateDenominator(
                1)

            .maxCaptures(10000000)

            .quality(80)

            .build())

    .build())

        .build()

            .build()

                .build();

        Map<String, AudioSelector> audioSelectors = new HashMap<>();
        audioSelectors.put("Audio Selector 1",

AudioSelector.builder().defaultSelection(AudioDefaultSelection.DEFAULT)
        .offset(0).build());

```

```
        JobSettings jobSettings =
JobSettings.builder().inputs(Input.builder()
                                .audioSelectors(audioSelectors)
                                .videoSelector(

VideoSelector.builder().colorSpace(ColorSpace.FOLLOW)

.rotate(InputRotate.DEGREE_0).build())

.filterEnable(InputFilterEnable.AUTO).filterStrength(0)
                                .deblockFilter(InputDeblockFilter.DISABLED)

.denoiseFilter(InputDenoiseFilter.DISABLED).psiControl(InputPsiControl.USE_PSI)

.timecodeSource(InputTimecodeSource.EMBEDDED).fileInput(fileInput).build())
                                .outputGroups(appleHLS, thumbs,
fileMp4).build();

        CreateJobRequest createJobRequest =
CreateJobRequest.builder().role(mcRoleARN)
                                .settings(jobSettings)
                                .build();

        CreateJobResponse createJobResponse =
emc.createJob(createJobRequest);
        return createJobResponse.job().id();

    } catch (MediaConvertException e) {
        System.out.println(e.toString());
        System.exit(0);
    }
    return "";
}

private final static Output createOutput(String customName,
        String nameModifier,
        String segmentModifier,
        int qvbrMaxBitrate,
        int qvbrQualityLevel,
        int originWidth,
        int originHeight,
        int targetWidth) {
```

```

        int targetHeight = Math.round(originHeight * targetWidth /
originWidth)
                                - (Math.round(originHeight * targetWidth /
originWidth) % 4);
        Output output = null;
        try {
            output =
Output.builder().nameModifier(nameModifier).outputSettings(OutputSettings.builder()
.hlsSettings(HlsSettings.builder().segmentModifier(segmentModifier)
.audioGroupId("program_audio")
.iFrameOnlyManifest(HlsIFrameOnlyManifest.EXCLUDE).build())
                .build())
.containerSettings(ContainerSettings.builder().container(ContainerType.M3_U8)
.m3u8Settings(M3u8Settings.builder().audioFramesPerPes(4)
.pcrControl(M3u8PcrControl.PCR_EVERY_PES_PACKET)
.pmtPid(480).privateMetadataPid(503)
.programNumber(1).patInterval(0).pmtInterval(0)
.scte35Source(M3u8Scte35Source.NONE)
.scte35Pid(500).nielsenId3(M3u8NielsenId3.NONE)
.timedMetadata(TimedMetadata.NONE)
.timedMetadataPid(502).videoPid(481)
.audioPids(482, 483, 484, 485, 486, 487, 488,
            489, 490, 491, 492)
                .build())
                .build()
                .videoDescription(
VideoDescription.builder().width(targetWidth)
.height(targetHeight)

```

```
.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(
    VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()
    .codec(VideoCodec.H_264)
    .h264Settings(H264Settings
        .builder()
        .rateControlMode(
            H264RateControlMode.QVBR)
        .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
        .qualityTuningLevel(
            H264QualityTuningLevel.SINGLE_PASS)
        .qvbrSettings(H264QvbrSettings
            .builder()
            .qvbrQualityLevel(
                qvbrQualityLevel)
            .build())
    .build())
```

```
.codecLevel(H264CodecLevel.AUTO)

.codecProfile((targetHeight > 720
              && targetWidth > 1280)
              ? H264CodecProfile.HIGH
              : H264CodecProfile.MAIN)

.maxBitrate(qvbrMaxBitrate)

.framerateControl(
              H264FramerateControl.INITIALIZE_FROM_SOURCE)

.gopSize(2.0)

.gopSizeUnits(H264GopSizeUnits.SECONDS)

.numberBFramesBetweenReferenceFrames(
              2)

.gopClosedCadence(
              1)

.gopBReference(H264GopBReference.DISABLED)

.slowPal(H264SlowPal.DISABLED)

.syntax(H264Syntax.DEFAULT)

.numberReferenceFrames(
              3)

.dynamicSubGop(H264DynamicSubGop.STATIC)

.fieldEncoding(H264FieldEncoding.PAFF)

.sceneChangeDetect(
```

```
                H264SceneChangeDetect.ENABLED)

        .minIInterval(0)

        .telecine(H264Telecine.NONE)

        .framerateConversionAlgorithm(

                H264FramerateConversionAlgorithm.DUPLICATE_DROP)

        .entropyEncoding(

                H264EntropyEncoding.CABAC)

        .slices(1)

        .unregisteredSeiTimecode(

                H264UnregisteredSeiTimecode.DISABLED)

        .repeatPps(H264RepeatPps.DISABLED)

        .adaptiveQuantization(

                H264AdaptiveQuantization.HIGH)

        .spatialAdaptiveQuantization(

                H264SpatialAdaptiveQuantization.ENABLED)

        .temporalAdaptiveQuantization(

                H264TemporalAdaptiveQuantization.ENABLED)

        .flickerAdaptiveQuantization(

                H264FlickerAdaptiveQuantization.DISABLED)

        .softness(0)

        .interlaceMode(H264InterlaceMode.PROGRESSIVE)

        .build())
```

```

        .build()
                                                    .build()

        .audioDescriptions(AudioDescription.builder()

        .audioTypeControl(AudioTypeControl.FOLLOW_INPUT)

        .languageCodeControl(AudioLanguageCodeControl.FOLLOW_INPUT)

        .codecSettings(AudioCodecSettings.builder()

        .codec(AudioCodec.AAC).aacSettings(AacSettings

        .builder()

        .codecProfile(AacCodecProfile.LC)

        .rateControlMode(

                AacRateControlMode.CBR)

        .codingMode(AacCodingMode.CODING_MODE_2_0)

        .sampleRate(44100)

        .bitrate(96000)

        .rawFormat(AacRawFormat.NONE)

        .specification(AacSpecification.MPEG4)

        .audioDescriptionBroadcasterMix(

                AacAudioDescriptionBroadcasterMix.NORMAL)

        .build()
                                                    .build()

                                                    .build()

                .build();
        } catch (MediaConvertException e) {
            e.printStackTrace();
            System.exit(0);
        }
    }

```

```
        return output;
    }
}
```

- Untuk detail API, lihat [CreateJob](#) di Referensi AWS SDK for Java 2.x API.

## Dapatkan pekerjaan transcoding

Contoh kode berikut menunjukkan cara mendapatkan pekerjaan AWS Elemental MediaConvert transcoding.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.GetJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.GetJobResponse;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import java.net.URI;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetJob {

    public static void main(String[] args) {

        final String usage = "\n" +
```



```
        " <jobId> \n\n" +
        "Where:\n" +
        " jobId - The job id value.\n\n";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String jobId = args[0];
    Region region = Region.US_WEST_2;
    MediaConvertClient mc = MediaConvertClient.builder()
        .region(region)
        .build();

    getSpecificJob(mc, jobId);
    mc.close();
}

public static void getSpecificJob(MediaConvertClient mc, String jobId) {
    try {
        DescribeEndpointsResponse res =
mc.describeEndpoints(DescribeEndpointsRequest.builder()
            .maxResults(20)
            .build());

        if (res.endpoints().size() <= 0) {
            System.out.println("Cannot find MediaConvert service endpoint
URL!");
            System.exit(1);
        }
        String endpointURL = res.endpoints().get(0).url();
        MediaConvertClient emc = MediaConvertClient.builder()
            .region(Region.US_WEST_2)
            .endpointOverride(URI.create(endpointURL))
            .build();

        GetJobRequest jobRequest = GetJobRequest.builder()
            .id(jobId)
            .build();

        GetJobResponse response = emc.getJob(jobRequest);
        System.out.println("The ARN of the job is " + response.job().arn());
    }
}
```

```
        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
}
```

- Untuk detail API, lihat [GetJob](#) di Referensi AWS SDK for Java 2.x API.

## Daftar pekerjaan transcoding

Contoh kode berikut menunjukkan bagaimana daftar pekerjaan AWS Elemental MediaConvert transcoding.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsResponse;
import software.amazon.awssdk.services.mediaconvert.model.Job;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import java.net.URI;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class ListJobs {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
            .region(region)
            .build();

        listCompleteJobs(mc);
        mc.close();
    }

    public static void listCompleteJobs(MediaConvertClient mc) {
        try {
            DescribeEndpointsResponse res =
mc.describeEndpoints(DescribeEndpointsRequest.builder()
                .maxResults(20)
                .build());

            if (res.endpoints().size() <= 0) {
                System.out.println("Cannot find MediaConvert service endpoint
URL!");
                System.exit(1);
            }

            String endpointURL = res.endpoints().get(0).url();
            MediaConvertClient emc = MediaConvertClient.builder()
                .region(Region.US_WEST_2)
                .endpointOverride(URI.create(endpointURL))
                .build();

            ListJobsRequest jobsRequest = ListJobsRequest.builder()
                .maxResults(10)
                .status("COMPLETE")
                .build();

            ListJobsResponse jobsResponse = emc.listJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("The JOB ARN is : " + job.arn());
            }

        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
}
```

```
    }  
  }  
}
```

- Untuk detail API, lihat [ListJobs](#) di Referensi AWS SDK for Java 2.x API.

## Contoh Migration Hub menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum menggunakan AWS SDK for Java 2.x with Migration Hub.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

### Tindakan

Hapus aliran kemajuan

Contoh kode berikut menunjukkan cara menghapus aliran kemajuan.

SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.DeleteProgressUpdateStreamRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteProgressStream {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <progressStream>\s

            Where:
                progressStream - the name of a progress stream to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String progressStream = args[0];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        deleteStream(migrationClient, progressStream);
        migrationClient.close();
    }

    public static void deleteStream(MigrationHubClient migrationClient, String
streamName) {
        try {
            DeleteProgressUpdateStreamRequest deleteProgressUpdateStreamRequest =
DeleteProgressUpdateStreamRequest
```

```
        .builder()
        .progressUpdateStreamName(streamName)
        .build();

migrationClient.deleteProgressUpdateStream(deleteProgressUpdateStreamRequest);
    System.out.println(streamName + " is deleted");


    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DeleteProgressUpdateStream](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan status migrasi

Contoh kode berikut menunjukkan cara menggambarkan status migrasi.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateRequest;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DescribeAppState {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                DescribeAppState <appId>\s

            Where:
                appId - the application id value.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        describeApplicationState(migrationClient, appId);
        migrationClient.close();
    }

    public static void describeApplicationState(MigrationHubClient migrationClient,
        String appId) {
        try {
            DescribeApplicationStateRequest applicationStateRequest =
            DescribeApplicationStateRequest.builder()
                .applicationId(appId)
                .build();

            DescribeApplicationStateResponse applicationStateResponse =
            migrationClient
                .describeApplicationState(applicationStateRequest);
            System.out.println("The application status is " +
            applicationStateResponse.applicationStatusAsString());
        }
    }
}
```

```

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}

```

- Untuk detail API, lihat [DescribeApplicationState](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan daftar atribut yang terkait dengan migrasi

Contoh kode berikut menunjukkan cara mendapatkan daftar atribut yang terkait dengan migrasi.

SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskRequest;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeMigrationTask {

    public static void main(String[] args) {
        final String usage = ""

```



```

Usage:
    DescribeMigrationTask <migrationTask> <progressStream>\s

Where:
    migrationTask - the name of a migration task.\s
    progressStream - the name of a progress stream.\s
""";

if (args.length < 2) {
    System.out.println(usage);
    System.exit(1);
}

String migrationTask = args[0];
String progressStream = args[1];
Region region = Region.US_WEST_2;
MigrationHubClient migrationClient = MigrationHubClient.builder()
    .region(region)
    .build();

describeMigTask(migrationClient, migrationTask, progressStream);
migrationClient.close();
}

public static void describeMigTask(MigrationHubClient migrationClient, String
migrationTask,
    String progressStream) {
    try {
        DescribeMigrationTaskRequest migrationTaskRequestRequest =
DescribeMigrationTaskRequest.builder()
            .progressUpdateStream(progressStream)
            .migrationTaskName(migrationTask)
            .build();

        DescribeMigrationTaskResponse migrationTaskResponse = migrationClient
            .describeMigrationTask(migrationTaskRequestRequest);
        System.out.println("The name is " +
migrationTaskResponse.migrationTask().migrationTaskName());

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

```

```
}  
}
```

- Untuk detail API, lihat [DescribeMigrationTask](#) di Referensi AWS SDK for Java 2.x API.

## Daftar aplikasi

Contoh kode berikut menunjukkan bagaimana daftar aplikasi.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;  
import software.amazon.awssdk.services.migrationhub.model.ApplicationState;  
import  
    software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesRequest;  
import  
    software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesResponse;  
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;  
import java.util.List;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class ListApplications {  
    public static void main(String[] args) {  
        Region region = Region.US_WEST_2;  
        MigrationHubClient migrationClient = MigrationHubClient.builder()  
            .region(region)  
            .build();
```

```
        listApps(migrationClient);
        migrationClient.close();
    }

    public static void listApps(MigrationHubClient migrationClient) {
        try {
            ListApplicationStatesRequest applicationStatesRequest =
                ListApplicationStatesRequest.builder()
                    .maxResults(10)
                    .build();

            ListApplicationStatesResponse response =
                migrationClient.listApplicationStates(applicationStatesRequest);
            List<ApplicationState> apps = response.applicationStateList();
            for (ApplicationState appState : apps) {
                System.out.println("App Id is " + appState.applicationId());
                System.out.println("The status is " +
                    appState.applicationStatus().toString());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListApplications](#) di Referensi AWS SDK for Java 2.x API.

## Daftar artefak yang dibuat

Contoh kode berikut menunjukkan cara membuat daftar artefak yang dibuat.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.CreatedArtifact;
import
    software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsRequest;
import
    software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCreatedArtifacts {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        listArtifacts(migrationClient);
        migrationClient.close();
    }

    public static void listArtifacts(MigrationHubClient migrationClient) {
        try {
            ListCreatedArtifactsRequest listCreatedArtifactsRequest =
                ListCreatedArtifactsRequest.builder()
                    .maxResults(10)
                    .migrationTaskName("SampleApp5")
                    .progressUpdateStream("ProgressSteamB")
                    .build();

            ListCreatedArtifactsResponse response =
                migrationClient.listCreatedArtifacts(listCreatedArtifactsRequest);
            List<CreatedArtifact> apps = response.createdArtifactList();
            for (CreatedArtifact artifact : apps) {
                System.out.println("App Id is " + artifact.description());
            }
        }
    }
}
```

```

        System.out.println("The name is " + artifact.name());
    }

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [ListCreatedArtifacts](#) di Referensi AWS SDK for Java 2.x API.

## Daftar tugas migrasi

Contoh kode berikut menunjukkan cara daftar tugas migrasi.

### SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksRequest;
import
    software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationTaskSummary;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

```

```
*/
public class ListMigrationTasks {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        listMigrTasks(migrationClient);
        migrationClient.close();
    }

    public static void listMigrTasks(MigrationHubClient migrationClient) {
        try {
            ListMigrationTasksRequest listMigrationTasksRequest =
ListMigrationTasksRequest.builder()
                .maxResults(10)
                .build();

            ListMigrationTasksResponse response =
migrationClient.listMigrationTasks(listMigrationTasksRequest);
            List<MigrationTaskSummary> migrationList =
response.migrationTaskSummaryList();
            for (MigrationTaskSummary migration : migrationList) {
                System.out.println("Migration task name is " +
migration.migrationTaskName());
                System.out.println("The Progress update stream is " +
migration.progressUpdateStream());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListMigrationTasks](#) di Referensi AWS SDK for Java 2.x API.

Daftarkan tugas migrasi

Contoh kode berikut menunjukkan cara mendaftarkan tugas migrasi.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.CreateProgressUpdateStreamRequest;
import
    software.amazon.awssdk.services.migrationhub.model.ImportMigrationTaskRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportMigrationTask {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <migrationTask> <progressStream>\s

                Where:
                migrationTask - the name of a migration task.\s
                progressStream - the name of a progress stream.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String migrationTask = args[0];
```

```
String progressStream = args[1];
Region region = Region.US_WEST_2;
MigrationHubClient migrationClient = MigrationHubClient.builder()
    .region(region)
    .build();

importMigrTask(migrationClient, migrationTask, progressStream);
migrationClient.close();
}

public static void importMigrTask(MigrationHubClient migrationClient, String
migrationTask, String progressStream) {
    try {
        CreateProgressUpdateStreamRequest progressUpdateStreamRequest =
CreateProgressUpdateStreamRequest.builder()
            .progressUpdateStreamName(progressStream)
            .dryRun(false)
            .build();

        migrationClient.createProgressUpdateStream(progressUpdateStreamRequest);
        ImportMigrationTaskRequest migrationTaskRequest =
ImportMigrationTaskRequest.builder()
            .migrationTaskName(migrationTask)
            .progressUpdateStream(progressStream)
            .dryRun(false)
            .build();

        migrationClient.importMigrationTask(migrationTaskRequest);

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [ImportMigrationTask](#) di Referensi AWS SDK for Java 2.x API.



## Amazon Personalisasi contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan Amazon Personalize. AWS SDK for Java 2.x

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

### Tindakan

Buat pekerjaan antarmuka batch

Contoh kode berikut menunjukkan cara membuat pekerjaan antarmuka batch Amazon Personalize.

SDK for Java 2.x

#### Note

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

```
public static String createPersonalizeBatchInferenceJob(PersonalizeClient
personalizeClient,
                String solutionVersionArn,
                String jobName,
                String s3InputDataSourcePath,
                String s3DataDestinationPath,
                String roleArn,
                String explorationWeight,
                String explorationItemAgeCutOff) {
```

```
    long waitInMilliseconds = 60 * 1000;
    String status;
    String batchInferenceJobArn;

    try {

        // Set up data input and output parameters.
        S3DataConfig inputSource = S3DataConfig.builder()
            .path(s3InputDataSourcePath)
            .build();

        S3DataConfig outputDestination = S3DataConfig.builder()
            .path(s3DataDestinationPath)
            .build();

        BatchInferenceJobInput jobInput =
BatchInferenceJobInput.builder()
            .s3DataSource(inputSource)
            .build();

        BatchInferenceJobOutput jobOutputLocation =
BatchInferenceJobOutput.builder()
            .s3DataDestination(outputDestination)
            .build();

        // Optional code to build the User-Personalization specific
item exploration
        // config.
        HashMap<String, String> explorationConfig = new HashMap<>();

        explorationConfig.put("explorationWeight",
explorationWeight);
        explorationConfig.put("explorationItemAgeCutOff",
explorationItemAgeCutOff);

        BatchInferenceJobConfig jobConfig =
BatchInferenceJobConfig.builder()
            .itemExplorationConfig(explorationConfig)
            .build();

        // End optional User-Personalization recipe specific code.
```

```

        CreateBatchInferenceJobRequest
createBatchInferenceJobRequest = CreateBatchInferenceJobRequest
        .builder()
        .solutionVersionArn(solutionVersionArn)
        .jobInput(jobInput)
        .jobOutput(jobOutputLocation)
        .jobName(jobName)
        .roleArn(roleArn)
        .batchInferenceJobConfig(jobConfig) //
Optional
        .build();

        batchInferenceJobArn =
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)
        .batchInferenceJobArn();

        DescribeBatchInferenceJobRequest
describeBatchInferenceJobRequest = DescribeBatchInferenceJobRequest
        .builder()
        .batchInferenceJobArn(batchInferenceJobArn)
        .build();

        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;
        while (Instant.now().getEpochSecond() < maxTime) {

                BatchInferenceJob batchInferenceJob =
personalizeClient
                .describeBatchInferenceJob(describeBatchInferenceJobRequest)
                        .batchInferenceJob();

                status = batchInferenceJob.status();
                System.out.println("Batch inference job status: " +
status);

                if (status.equals("ACTIVE") || status.equals("CREATE
FAILED")) {

                        break;
                }
                try {
                        Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                        System.out.println(e.getMessage());
                }

```

```
        }
        return batchInferenceJobArn;

    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

- Untuk detail API, lihat [CreateBatchInferenceJob](#) di Referensi AWS SDK for Java 2.x API.

## Buat kampanye

Contoh kode berikut menunjukkan cara membuat kampanye Amazon Personalize.

### SDK for Java 2.x

#### Note

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

```
public static void createPersonalCampaign(PersonalizeClient personalizeClient,
String solutionVersionArn,
String name) {

    try {
        CreateCampaignRequest createCampaignRequest =
CreateCampaignRequest.builder()
            .minProvisionedTPS(1)
            .solutionVersionArn(solutionVersionArn)
            .name(name)
            .build();

        CreateCampaignResponse campaignResponse =
personalizeClient.createCampaign(createCampaignRequest);
        System.out.println("The campaign ARN is " +
campaignResponse.campaignArn());
    }
}
```

```
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Untuk detail API, lihat [CreateCampaign](#) di Referensi AWS SDK for Java 2.x API.

## Buat kumpulan data

Contoh kode berikut menunjukkan cara membuat dataset Amazon Personalize.

### SDK for Java 2.x

#### Note

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

```
public static String createDataset(PersonalizeClient personalizeClient,  
    String datasetName,  
    String datasetGroupArn,  
    String datasetType,  
    String schemaArn) {  
    try {  
        CreateDatasetRequest request = CreateDatasetRequest.builder()  
            .name(datasetName)  
            .datasetGroupArn(datasetGroupArn)  
            .datasetType(datasetType)  
            .schemaArn(schemaArn)  
            .build();  
  
        String datasetArn = personalizeClient.createDataset(request)  
            .datasetArn();  
        System.out.println("Dataset " + datasetName + " created.");  
        return datasetArn;  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
    }  
}
```

```
        System.exit(1);
    }
    return "";
}
```

- Untuk detail API, lihat [CreateDataset](#) di Referensi AWS SDK for Java 2.x API.

## Buat pekerjaan ekspor dataset

Contoh kode berikut menunjukkan cara membuat pekerjaan ekspor dataset Amazon Personalize.

### SDK for Java 2.x

#### Note

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

```
public static String createDatasetExportJob(PersonalizeClient personalizeClient,
    String jobName,
    String datasetArn,
    IngestionMode ingestionMode,
    String roleArn,
    String s3BucketPath,
    String kmsKeyArn) {

    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String status = null;

    try {

        S3DataConfig exportS3DataConfig =
        S3DataConfig.builder().path(s3BucketPath).kmsKeyArn(kmsKeyArn).build();
        DatasetExportJobOutput jobOutput =
        DatasetExportJobOutput.builder().s3DataDestination(exportS3DataConfig)
            .build();

        CreateDatasetExportJobRequest createRequest =
        CreateDatasetExportJobRequest.builder()
```

```
        .jobName(jobName)
        .datasetArn(datasetArn)
        .ingestionMode(ingestionMode)
        .jobOutput(jobOutput)
        .roleArn(roleArn)
        .build();

    String datasetExportJobArn =
personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

    DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
DescribeDatasetExportJobRequest.builder()
        .datasetExportJobArn(datasetExportJobArn)
        .build();

    long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

    while (Instant.now().getEpochSecond() < maxTime) {

        DatasetExportJob datasetExportJob = personalizeClient
            .describeDatasetExportJob(describeDatasetExportJobRequest)
            .datasetExportJob();

        status = datasetExportJob.status();
        System.out.println("Export job status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            return status;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}
```

- Untuk detail API, lihat [CreateDatasetExportJob](#) di Referensi AWS SDK for Java 2.x API.

## Membuat grup dataset

Contoh kode berikut menunjukkan cara membuat grup dataset Amazon Personalize.

SDK for Java 2.x

### Note

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

```
public static String createDatasetGroup(PersonalizeClient personalizeClient,
String datasetGroupName) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
            .name(datasetGroupName)
            .build();

        return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

Buat grup dataset domain.

```
public static String createDomainDatasetGroup(PersonalizeClient
personalizeClient,
        String datasetGroupName,
        String domain) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
            .name(datasetGroupName)
            .domain(domain)
            .build();
```



```

        return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}

```

- Untuk detail API, lihat [CreateDatasetGroup](#) di Referensi AWS SDK for Java 2.x API.

## Buat pekerjaan impor dataset

Contoh kode berikut menunjukkan cara membuat pekerjaan impor dataset Amazon Personalize.

### SDK for Java 2.x

#### Note

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

```

public static String createPersonalizeDatasetImportJob(PersonalizeClient
personalizeClient,
    String jobName,
    String datasetArn,
    String s3BucketPath,
    String roleArn) {

    long waitInMilliseconds = 60 * 1000;
    String status;
    String datasetImportJobArn;

    try {
        DataSource importDataSource = DataSource.builder()
            .dataLocation(s3BucketPath)
            .build();

        CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()

```

```

        .datasetArn(datasetArn)
        .dataSource(importDataSource)
        .jobName(jobName)
        .roleArn(roleArn)
        .build();

    datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
        .datasetImportJobArn();
    DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
        .datasetImportJobArn(datasetImportJobArn)
        .build();

    long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

    while (Instant.now().getEpochSecond() < maxTime) {

        DatasetImportJob datasetImportJob = personalizeClient
            .describeDatasetImportJob(describeDatasetImportJobRequest)
            .datasetImportJob();

        status = datasetImportJob.status();
        System.out.println("Dataset import job status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return datasetImportJobArn;

} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}

```

- Untuk detail API, lihat [CreateDatasetImportJob](#) di Referensi AWS SDK for Java 2.x API.

## Buat skema domain

Contoh kode berikut menunjukkan cara membuat skema domain Amazon Personalize.

### SDK for Java 2.x

#### Note

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

```
public static String createDomainSchema(PersonalizeClient personalizeClient,
String schemaName, String domain,
String filePath) {

String schema = null;
try {
schema = new String(Files.readAllBytes(Paths.get(filePath)));
} catch (IOException e) {
System.out.println(e.getMessage());
}

try {
CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
.name(schemaName)
.domain(domain)
.schema(schema)
.build();

String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

System.out.println("Schema arn: " + schemaArn);

return schemaArn;

} catch (PersonalizeException e) {
System.err.println(e.awsErrorDetails().errorMessage());
System.exit(1);
}
return "";
}
```

- Untuk detail API, lihat [CreateSchema](#) di Referensi AWS SDK for Java 2.x API.

## Buat filter

Contoh kode berikut menunjukkan cara membuat filter Amazon Personalize.

## SDK for Java 2.x

### Note

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

```
public static String createFilter(PersonalizeClient personalizeClient,
    String filterName,
    String datasetGroupArn,
    String filterExpression) {
    try {
        CreateFilterRequest request = CreateFilterRequest.builder()
            .name(filterName)
            .datasetGroupArn(datasetGroupArn)
            .filterExpression(filterExpression)
            .build();

        return personalizeClient.createFilter(request).filterArn();
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Untuk detail API, lihat [CreateFilter](#) di Referensi AWS SDK for Java 2.x API.

## Buat pemberi rekomendasi

Contoh kode berikut menunjukkan cara membuat pemberi rekomendasi Amazon Personalize.

## SDK for Java 2.x

 Note

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

```
public static String createRecommender(PersonalizeClient personalizeClient,
    String name,
    String datasetGroupArn,
    String recipeArn) {

    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String recommenderStatus = "";

    try {
        CreateRecommenderRequest createRecommenderRequest =
CreateRecommenderRequest.builder()
            .datasetGroupArn(datasetGroupArn)
            .name(name)
            .recipeArn(recipeArn)
            .build();

        CreateRecommenderResponse recommenderResponse = personalizeClient
            .createRecommender(createRecommenderRequest);
        String recommenderArn = recommenderResponse.recommenderArn();
        System.out.println("The recommender ARN is " + recommenderArn);

        DescribeRecommenderRequest describeRecommenderRequest =
DescribeRecommenderRequest.builder()
            .recommenderArn(recommenderArn)
            .build();

        maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        while (Instant.now().getEpochSecond() < maxTime) {

            recommenderStatus =
personalizeClient.describeRecommender(describeRecommenderRequest).recommender()
                .status();
        }
    }
}
```

```
        System.out.println("Recommender status: " + recommenderStatus);

        if (recommenderStatus.equals("ACTIVE") ||
recommenderStatus.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return recommenderArn;

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

- Untuk detail API, lihat [CreateRecommender](#) di Referensi AWS SDK for Java 2.x API.

## Buat skema

Contoh kode berikut menunjukkan cara membuat skema Amazon Personalize.

## SDK for Java 2.x

### Note

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

```
public static String createSchema(PersonalizeClient personalizeClient, String
schemaName, String filePath) {

    String schema = null;
    try {
        schema = new String(Files.readAllBytes(Paths.get(filePath)));
    }
```

```
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }

    try {
        CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
            .name(schemaName)
            .schema(schema)
            .build();

        String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

        System.out.println("Schema arn: " + schemaArn);

        return schemaArn;


    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Untuk detail API, lihat [CreateSchema](#) di Referensi AWS SDK for Java 2.x API.

Buat solusi

Contoh kode berikut menunjukkan cara membuat solusi Amazon Personalize.

SDK for Java 2.x

 Note

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

```
public static String createPersonalizeSolution(PersonalizeClient
personalizeClient,
        String datasetGroupArn,
```

```

        String solutionName,
        String recipeArn) {

    try {
        CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
            .name(solutionName)
            .datasetGroupArn(datasetGroupArn)
            .recipeArn(recipeArn)
            .build();

        CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);
        return solutionResponse.solutionArn();

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

```

- Untuk detail API, lihat [CreateSolution](#) di Referensi AWS SDK for Java 2.x API.

Buat versi solusi

Contoh kode berikut menunjukkan cara membuat solusi Amazon Personalize.

SDK for Java 2.x

#### Note

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

```

public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String solutionStatus = "";
    String solutionVersionStatus = "";

```



```
String solutionVersionArn = "";

try {
    DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
        .solutionArn(solutionArn)
        .build();

    maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

    // Wait until solution is active.
    while (Instant.now().getEpochSecond() < maxTime) {

        solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
        System.out.println("Solution status: " + solutionStatus);

        if (solutionStatus.equals("ACTIVE") || solutionStatus.equals("CREATE
FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }

    if (solutionStatus.equals("ACTIVE")) {

        CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient
            .createSolutionVersion(createSolutionVersionRequest);
        solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

        System.out.println("Solution version ARN: " + solutionVersionArn);
    }
}
```

```

        DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
        .solutionVersionArn(solutionVersionArn)
        .build();

        while (Instant.now().getEpochSecond() < maxTime) {

            solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest)
                .solutionVersion().status();
            System.out.println("Solution version status: " +
solutionVersionStatus);

            if (solutionVersionStatus.equals("ACTIVE") ||
solutionVersionStatus.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
        return solutionVersionArn;
    }
} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

```

- Untuk detail API, lihat [CreateSolutionVersion](#) di Referensi AWS SDK for Java 2.x API.

Buat pelacak acara

Contoh kode berikut menunjukkan cara membuat pelacak peristiwa Amazon Personalize.

## SDK for Java 2.x

 Note

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

```
public static String createEventTracker(PersonalizeClient personalizeClient,
String eventTrackerName,
    String datasetGroupArn) {

    String eventTrackerId = "";
    String eventTrackerArn;
    long maxTime = 3 * 60 * 60; // 3 hours
    long waitInMilliseconds = 20 * 1000; // 20 seconds
    String status;

    try {

        CreateEventTrackerRequest createEventTrackerRequest =
CreateEventTrackerRequest.builder()
            .name(eventTrackerName)
            .datasetGroupArn(datasetGroupArn)
            .build();

        CreateEventTrackerResponse createEventTrackerResponse =
personalizeClient
            .createEventTracker(createEventTrackerRequest);

        eventTrackerArn = createEventTrackerResponse.eventTrackerArn();
        eventTrackerId = createEventTrackerResponse.trackingId();
        System.out.println("Event tracker ARN: " + eventTrackerArn);
        System.out.println("Event tracker ID: " + eventTrackerId);

        maxTime = Instant.now().getEpochSecond() + maxTime;

        DescribeEventTrackerRequest describeRequest =
DescribeEventTrackerRequest.builder()
            .eventTrackerArn(eventTrackerArn)
            .build();
```

```
while (Instant.now().getEpochSecond() < maxTime) {  
  
    status =  
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();  
    System.out.println("EventTracker status: " + status);  
  
    if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {  
        break;  
    }  
    try {  
        Thread.sleep(waitInMilliseconds);  
    } catch (InterruptedException e) {  
        System.out.println(e.getMessage());  
    }  
}  
return eventTrackerId;  
} catch (PersonalizeException e) {  
    System.out.println(e.awsErrorDetails().errorMessage());  
    System.exit(1);  
}  
return eventTrackerId;  
}
```

- Untuk detail API, lihat [CreateEventTracker](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus kampanye

Contoh kode berikut menunjukkan cara menghapus kampanye di Amazon Personalize.

SDK for Java 2.x

### Note

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

```
public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,  
String campaignArn) {
```

```
try {
    DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()
        .campaignArn(campaignArn)
        .build();

    personalizeClient.deleteCampaign(campaignRequest);

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Untuk detail API, lihat [DeleteCampaign](#) di Referensi AWS SDK for Java 2.x API.

## Hapus solusi

Contoh kode berikut menunjukkan cara menghapus solusi di Amazon Personalize.

### SDK for Java 2.x

#### Note

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

```
public static void deleteGivenSolution(PersonalizeClient personalizeClient,
String solutionArn) {

    try {
        DeleteSolutionRequest solutionRequest = DeleteSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        personalizeClient.deleteSolution(solutionRequest);
        System.out.println("Done");

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DeleteSolution](#) di Referensi AWS SDK for Java 2.x API.

## Hapus pelacak acara

Contoh kode berikut menunjukkan cara menghapus pelacak peristiwa di Amazon Personalize.

### SDK for Java 2.x

#### Note

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

```
public static void deleteEventTracker(PersonalizeClient personalizeClient,
String eventTrackerArn) {
    try {
        DeleteEventTrackerRequest deleteEventTrackerRequest =
DeleteEventTrackerRequest.builder()
            .eventTrackerArn(eventTrackerArn)
            .build();

        int status =
personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().statusCode();

        System.out.println("Status code:" + status);

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DeleteEventTracker](#) di Referensi AWS SDK for Java 2.x API.

## Jelaskan kampanye

Contoh kode berikut menunjukkan cara mendeskripsikan kampanye di Amazon Personalize.

SDK for Java 2.x

### Note

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

```
public static void describeSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {

    try {
        DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
        Campaign myCampaign = campaignResponse.campaign();
        System.out.println("The Campaign name is " + myCampaign.name());
        System.out.println("The Campaign status is " + myCampaign.status());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DescribeCampaign](#) di Referensi AWS SDK for Java 2.x API.

## Jelaskan resepnya

Contoh kode berikut menunjukkan cara mendeskripsikan resep di Amazon Personalize.

## SDK for Java 2.x

 Note

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

```
public static void describeSpecificRecipe(PersonalizeClient personalizeClient,
String recipeArn) {

    try {
        DescribeRecipeRequest recipeRequest = DescribeRecipeRequest.builder()
            .recipeArn(recipeArn)
            .build();

        DescribeRecipeResponse recipeResponse =
personalizeClient.describeRecipe(recipeRequest);
        System.out.println("The recipe name is " +
recipeResponse.recipe().name());


    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DescribeRecipe](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan solusinya

Contoh kode berikut menunjukkan cara mendeskripsikan solusi di Amazon Personalize.

## SDK for Java 2.x

 Note

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



```
public static void describeSpecificSolution(PersonalizeClient personalizeClient,
String solutionArn) {

    try {
        DescribeSolutionRequest solutionRequest =
DescribeSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        DescribeSolutionResponse response =
personalizeClient.describeSolution(solutionRequest);
        System.out.println("The Solution name is " +
response.solution().name());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DescribeSolution](#) di Referensi AWS SDK for Java 2.x API.

## Daftar kampanye

Contoh kode berikut menunjukkan cara membuat daftar kampanye di Amazon Personalize.

### SDK for Java 2.x

#### Note

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

```
public static void listAllCampaigns(PersonalizeClient personalizeClient, String
solutionArn) {

    try {
        ListCampaignsRequest campaignsRequest = ListCampaignsRequest.builder()
            .maxResults(10)
```

```

        .solutionArn(solutionArn)
        .build();

    ListCampaignsResponse response =
personalizeClient.listCampaigns(campaignsRequest);
    List<CampaignSummary> campaigns = response.campaigns();
    for (CampaignSummary campaign : campaigns) {
        System.out.println("Campaign name is : " + campaign.name());
        System.out.println("Campaign ARN is : " + campaign.campaignArn());
    }

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

```

- Untuk detail API, lihat [ListCampaigns](#) di Referensi AWS SDK for Java 2.x API.

## Daftar grup kumpulan data

Contoh kode berikut menunjukkan cara mencantumkan grup kumpulan data di Amazon Personalize.

### SDK for Java 2.x

#### Note

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

```

public static void listDSGroups(PersonalizeClient personalizeClient) {

    try {
        ListDatasetGroupsRequest groupsRequest =
ListDatasetGroupsRequest.builder()
            .maxResults(15)
            .build();

        ListDatasetGroupsResponse groupsResponse =
personalizeClient.listDatasetGroups(groupsRequest);
    }
}

```

```
List<DatasetGroupSummary> groups = groupsResponse.datasetGroups();
for (DatasetGroupSummary group : groups) {
    System.out.println("The DataSet name is : " + group.name());
    System.out.println("The DataSet ARN is : " +
group.datasetGroupArn());
}

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Untuk detail API, lihat [ListDatasetGroups](#) di Referensi AWS SDK for Java 2.x API.

## Daftar resep

Contoh kode berikut menunjukkan cara membuat daftar resep di Amazon Personalize.

## SDK for Java 2.x

### Note

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

```
public static void listAllRecipes(PersonalizeClient personalizeClient) {

    try {
        ListRecipesRequest recipesRequest = ListRecipesRequest.builder()
            .maxResults(15)
            .build();

        ListRecipesResponse response =
personalizeClient.listRecipes(recipesRequest);
        List<RecipeSummary> recipes = response.recipes();
        for (RecipeSummary recipe : recipes) {
            System.out.println("The recipe ARN is: " + recipe.recipeArn());
            System.out.println("The recipe name is: " + recipe.name());
        }
    }
}
```

```
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ListRecipes](#) di Referensi AWS SDK for Java 2.x API.

## Daftar solusi

Contoh kode berikut menunjukkan cara membuat daftar solusi di Amazon Personalize.

## SDK for Java 2.x

### Note

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

```
public static void listAllSolutions(PersonalizeClient personalizeClient, String
datasetGroupArn) {

    try {
        ListSolutionsRequest solutionsRequest = ListSolutionsRequest.builder()
            .maxResults(10)
            .datasetGroupArn(datasetGroupArn)
            .build();

        ListSolutionsResponse response =
personalizeClient.listSolutions(solutionsRequest);
        List<SolutionSummary> solutions = response.solutions();
        for (SolutionSummary solution : solutions) {
            System.out.println("The solution ARN is: " +
solution.solutionArn());
            System.out.println("The solution name is: " + solution.name());
        }

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ListSolutions](#) di Referensi AWS SDK for Java 2.x API.

## Memperbarui kampanye

Contoh kode berikut menunjukkan cara memperbarui kampanye Amazon Personalize.

### SDK for Java 2.x

#### Note

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

```
public static String updateCampaign(PersonalizeClient personalizeClient,
    String campaignArn,
    String solutionVersionArn,
    Integer minProvisionedTPS) {

    try {
        // build the updateCampaignRequest
        UpdateCampaignRequest updateCampaignRequest =
UpdateCampaignRequest.builder()
            .campaignArn(campaignArn)
            .solutionVersionArn(solutionVersionArn)
            .minProvisionedTPS(minProvisionedTPS)
            .build();

        // update the campaign
        personalizeClient.updateCampaign(updateCampaignRequest);

        DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();
```

```
        DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
        Campaign updatedCampaign = campaignResponse.campaign();

        System.out.println("The Campaign status is " +
updatedCampaign.status());
        return updatedCampaign.status();

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Untuk detail API, lihat [UpdateCampaign](#) di Referensi AWS SDK for Java 2.x API.

## Amazon Personalize Events contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum AWS SDK for Java 2.x dengan menggunakan Peristiwa Personalisasi Amazon.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

## Tindakan

### Impor data peristiwa interaksi waktu nyata

Contoh kode berikut menunjukkan cara mengimpor data peristiwa interaksi real-time ke Amazon Personalize Events.

### SDK for Java 2.x

#### Note

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

```
public static int putItems(PersonalizeEventsClient personalizeEventsClient,
    String datasetArn,
    String item1Id,
    String item1PropertyName,
    String item1PropertyValue,
    String item2Id,
    String item2PropertyName,
    String item2PropertyValue) {

    int responseCode = 0;
    ArrayList<Item> items = new ArrayList<>();

    try {
        Item item1 = Item.builder()
            .itemId(item1Id)
            .properties(String.format("{ \"%1$s\": \"%2$s
\}"}",
                item1PropertyName,
                item1PropertyValue))
            .build();

        items.add(item1);

        Item item2 = Item.builder()
            .itemId(item2Id)
            .properties(String.format("{ \"%1$s\": \"%2$s
\}"}",
```

```
        item2PropertyName,
        item2PropertyValue))
            .build();

        items.add(item2);

        PutItemsRequest putItemsRequest = PutItemsRequest.builder()
            .datasetArn(datasetArn)
            .items(items)
            .build();

        responseCode =
personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;


    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return responseCode;
}
```

- Untuk detail API, lihat [PutEvents](#) di Referensi AWS SDK for Java 2.x API.

Impor pengguna secara bertahap

Contoh kode berikut menunjukkan cara mengimpor pengguna secara bertahap ke Amazon Personalize Events Events.

SDK for Java 2.x

 Note

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

```
public static int putUsers(PersonalizeEventsClient personalizeEventsClient,
    String datasetArn,
    String user1Id,
    String user1PropertyName,
```



```
        String user1PropertyValue,
        String user2Id,
        String user2PropertyName,
        String user2PropertyValue) {

    int responseCode = 0;
    ArrayList<User> users = new ArrayList<>();

    try {
        User user1 = User.builder()
            .userId(user1Id)
            .properties(String.format("{\\"%1$s\\": \\"%2$s
\\"}",
                                user1PropertyName,
                                user1PropertyValue))
            .build();

        users.add(user1);

        User user2 = User.builder()
            .userId(user2Id)
            .properties(String.format("{\\"%1$s\\": \\"%2$s
\\"}",
                                user2PropertyName,
                                user2PropertyValue))
            .build();

        users.add(user2);

        PutUsersRequest putUsersRequest = PutUsersRequest.builder()
            .datasetArn(datasetArn)
            .users(users)
            .build();

        responseCode =
personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;

    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return responseCode;
}
```

- Untuk detail API, lihat [PutUsers](#) di Referensi AWS SDK for Java 2.x API.

## Amazon Personalisasi contoh Runtime menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum menggunakan Runtime AWS SDK for Java 2.x with Amazon Personalize.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

### Tindakan

Dapatkan rekomendasi (grup dataset kustom)

Contoh kode berikut menunjukkan cara mendapatkan rekomendasi peringkat Amazon Personalize Runtime Runtime.

SDK for Java 2.x

#### Note

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

```
public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
```

```

        String campaignArn,
        String userId,
        ArrayList<String> items) {

    try {
        GetPersonalizedRankingRequest rankingRecommendationsRequest =
        GetPersonalizedRankingRequest.builder()
            .campaignArn(campaignArn)
            .userId(userId)
            .inputList(items)
            .build();

        GetPersonalizedRankingResponse recommendationsResponse =
        personalizeRuntimeClient
            .getPersonalizedRanking(rankingRecommendationsRequest);
        List<PredictedItem> rankedItems =
        recommendationsResponse.personalizedRanking();
        int rank = 1;
        for (PredictedItem item : rankedItems) {
            System.out.println("Item ranked at position " + rank + " details");
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
            System.out.println("-----");
            rank++;
        }
        return rankedItems;
    } catch (PersonalizeRuntimeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

```

- Untuk detail API, lihat [GetPersonalizedRanking](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan rekomendasi dari pemberi rekomendasi (grup dataset domain)

Contoh kode berikut menunjukkan cara mendapatkan rekomendasi Amazon Personalize Runtime.

## SDK for Java 2.x

 Note

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

Dapatkan daftar item yang direkomendasikan.

```
public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String campaignArn, String userId) {

    try {
        GetRecommendationsRequest recommendationsRequest =
        GetRecommendationsRequest.builder()
            .campaignArn(campaignArn)
            .numResults(20)
            .userId(userId)
            .build();

        GetRecommendationsResponse recommendationsResponse =
        personalizeRuntimeClient
            .getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();
        for (PredictedItem item : items) {
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
        }

    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Dapatkan daftar item yang direkomendasikan dari pemberi rekomendasi yang dibuat dalam grup kumpulan data domain.

```
public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String recommenderArn,
```

```
        String userId) {

    try {
        GetRecommendationsRequest recommendationsRequest =
        GetRecommendationsRequest.builder()
            .recommenderArn(recommenderArn)
            .numResults(20)
            .userId(userId)
            .build();

        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
            .getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();

        for (PredictedItem item : items) {
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
        }
    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Gunakan filter saat meminta rekomendasi.

```
public static void getFilteredRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
    String campaignArn,
    String userId,
    String filterArn,
    String parameter1Name,
    String parameter1Value1,
    String parameter1Value2,
    String parameter2Name,
    String parameter2Value) {

    try {

        Map<String, String> filterValues = new HashMap<>();
```

```
filterValues.put(parameter1Name, String.format("\"%1$s\", \"%2$s\"",
    parameter1Value1, parameter1Value2));
filterValues.put(parameter2Name, String.format("\"%1$s\"",
    parameter2Value));

GetRecommendationsRequest recommendationsRequest =
personalizeRuntimeClient
    .campaignArn(campaignArn)
    .numResults(20)
    .userId(userId)
    .filterArn(filterArn)
    .filterValues(filterValues)
    .build();

GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
    .getRecommendations(recommendationsRequest);
List<PredictedItem> items = recommendationsResponse.itemList();

for (PredictedItem item : items) {
    System.out.println("Item Id is : " + item.itemId());
    System.out.println("Item score is : " + item.score());
}
} catch (PersonalizeRuntimeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Untuk detail API, lihat [GetRecommendations](#) di Referensi AWS SDK for Java 2.x API.

## Amazon Pinpoint contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x dengan Amazon Pinpoint.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

## Tindakan

Buat kampanye

Contoh kode berikut menunjukkan cara membuat kampanye.

SDK for Java 2.x

### Note

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

Buat kampanye.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.Message;
import software.amazon.awssdk.services.pinpoint.model.Schedule;
import software.amazon.awssdk.services.pinpoint.model.Action;
import software.amazon.awssdk.services.pinpoint.model.MessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.WriteCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateCampaign {
    public static void main(String[] args) {

        final String usage = ""

            Usage:  <appId> <segmentId>

            Where:
                appId - The ID of the application to create the campaign in.
                segmentId - The ID of the segment to create the campaign from.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String segmentId = args[1];
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createPinCampaign(pinpoint, appId, segmentId);
        pinpoint.close();
    }

    public static void createPinCampaign(PinpointClient pinpoint, String appId,
String segmentId) {
        CampaignResponse result = createCampaign(pinpoint, appId, segmentId);
        System.out.println("Campaign " + result.name() + " created.");
        System.out.println(result.description());
    }

    public static CampaignResponse createCampaign(PinpointClient client, String
appId, String segmentID) {

        try {
            Schedule schedule = Schedule.builder()
                .startTime("IMMEDIATE")
```



```
        .build();

        Message defaultMessage = Message.builder()
            .action(Action.OPEN_APP)
            .body("My message body.")
            .title("My message title.")
            .build();

        MessageConfiguration messageConfiguration =
MessageConfiguration.builder()
            .defaultMessage(defaultMessage)
            .build();

        WriteCampaignRequest request = WriteCampaignRequest.builder()
            .description("My description")
            .schedule(schedule)
            .name("MyCampaign")
            .segmentId(segmentID)
            .messageConfiguration(messageConfiguration)
            .build();

        CreateCampaignResponse result =
client.createCampaign(CreateCampaignRequest.builder()
            .applicationId(appID)
            .writeCampaignRequest(request).build());

        System.out.println("Campaign ID: " + result.campaignResponse().id());
        return result.campaignResponse();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
}
```

- Untuk detail API, lihat [CreateCampaign](#) di Referensi AWS SDK for Java 2.x API.

## Membuat sebuah segmen

Contoh kode berikut menunjukkan cara membuat segmen.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AttributeDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.AttributeType;
import software.amazon.awssdk.services.pinpoint.model.RecencyDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentBehaviors;
import software.amazon.awssdk.services.pinpoint.model.SegmentDemographics;
import software.amazon.awssdk.services.pinpoint.model.SegmentLocation;
import software.amazon.awssdk.services.pinpoint.model.SegmentDimensions;
import software.amazon.awssdk.services.pinpoint.model.WriteSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateSegment {
    public static void main(String[] args) {
        final String usage = ""

                                Usage:  <appId>
```

```

        Where:
            appId - The application ID to create a segment
for.

        """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        SegmentResponse result = createSegment(pinpoint, appId);
        System.out.println("Segment " + result.name() + " created.");
        System.out.println(result.segmentType());
        pinpoint.close();
    }

    public static SegmentResponse createSegment(PinpointClient client, String
appId) {
        try {
            Map<String, AttributeDimension> segmentAttributes = new
HashMap<>();

            segmentAttributes.put("Team", AttributeDimension.builder()
                .attributeType(AttributeType.INCLUSIVE)
                .values("Lakers")
                .build());

            RecencyDimension recencyDimension =
RecencyDimension.builder()
                .duration("DAY_30")
                .recencyType("ACTIVE")
                .build();

            SegmentBehaviors segmentBehaviors =
SegmentBehaviors.builder()
                .recency(recencyDimension)
                .build();

```

```
SegmentDemographics segmentDemographics =
SegmentDemographics
    .builder()
    .build();

SegmentLocation segmentLocation = SegmentLocation
    .builder()
    .build();

SegmentDimensions dimensions = SegmentDimensions
    .builder()
    .attributes(segmentAttributes)
    .behavior(segmentBehaviors)
    .demographic(segmentDemographics)
    .location(segmentLocation)
    .build();

WriteSegmentRequest writeSegmentRequest =
WriteSegmentRequest.builder()
    .name("MySegment")
    .dimensions(dimensions)
    .build();

CreateSegmentRequest createSegmentRequest =
CreateSegmentRequest.builder()
    .applicationId(appId)
    .writeSegmentRequest(writeSegmentRequest)
    .build();

CreateSegmentResponse createSegmentResult =
client.createSegment(createSegmentRequest);
System.out.println("Segment ID: " +
createSegmentResult.segmentResponse().id());
System.out.println("Done");
return createSegmentResult.segmentResponse();

} catch (PinpointException e) {
System.err.println(e.awsErrorDetails().errorMessage());
System.exit(1);
}
return null;
}
}
```

- Untuk detail API, lihat [CreateSegment](#) di Referensi AWS SDK for Java 2.x API.

## Membuat aplikasi

Contoh kode berikut menunjukkan cara membuat aplikasi.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateAppRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateAppResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateApplicationRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateApp {
    public static void main(String[] args) {
        final String usage = ""

            Usage: <appName>

            Where:
                appName - The name of the application to create.

        """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        String appName = args[0];
        System.out.println("Creating an application with name: " + appName);

        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String appID = createApplication(pinpoint, appName);
        System.out.println("App ID is: " + appID);
        pinpoint.close();
    }

    public static String createApplication(PinpointClient pinpoint, String appName)
    {
        try {
            CreateApplicationRequest appRequest = CreateApplicationRequest.builder()
                .name(appName)
                .build();

            CreateAppRequest request = CreateAppRequest.builder()
                .createApplicationRequest(appRequest)
                .build();

            CreateAppResponse result = pinpoint.createApp(request);
            return result.applicationResponse().id();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Untuk detail API, lihat [CreateApp](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus sebuah aplikasi

Contoh kode berikut menunjukkan cara menghapus aplikasi.

### SDK for Java 2.x

#### Note

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

### Hapus aplikasi.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteApp {
    public static void main(String[] args) {
        final String usage = ""

                Usage: <appId>

                Where:
                appId - The ID of the application to delete.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String appId = args[0];
System.out.println("Deleting an application with ID: " + appId);
PinpointClient pinpoint = PinpointClient.builder()
    .region(Region.US_EAST_1)
    .build();

deletePinApp(pinpoint, appId);
System.out.println("Done");
pinpoint.close();
}

public static void deletePinApp(PinpointClient pinpoint, String appId) {
    try {
        DeleteAppRequest appRequest = DeleteAppRequest.builder()
            .applicationId(appId)
            .build();

        DeleteAppResponse result = pinpoint.deleteApp(appRequest);
        String appName = result.applicationResponse().name();
        System.out.println("Application " + appName + " has been deleted.");

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DeleteApp](#) di Referensi AWS SDK for Java 2.x API.

## Hapus titik akhir

Contoh kode berikut menunjukkan cara menghapus titik akhir.

## SDK for Java 2.x

### Note

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



## Hapus titik akhir.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteEndpoint {
    public static void main(String[] args) {
        final String usage = ""

                Usage:  <appName> <endpointId >

                Where:
                    appId - The id of the application to delete.
                    endpointId - The id of the endpoint to delete.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpointId = args[1];
        System.out.println("Deleting an endpoint with id: " + endpointId);
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        deletePinEncpoint(pinpoint, appId, endpointId);
        pinpoint.close();
    }
}
```

```
public static void deletePinEndpoint(PinpointClient pinpoint, String appId,
String endpointId) {
    try {
        DeleteEndpointRequest appRequest = DeleteEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpointId)
            .build();

        DeleteEndpointResponse result = pinpoint.deleteEndpoint(appRequest);
        String id = result.endpointResponse().id();
        System.out.println("The deleted endpoint id " + id);

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

- Untuk detail API, lihat [DeleteEndpoint](#) di Referensi AWS SDK for Java 2.x API.

## Ekspor titik akhir

Contoh kode berikut menunjukkan cara mengekspor titik akhir.

## SDK for Java 2.x

### Note

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

## Ekspor titik akhir.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.ExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
```

```
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobRequest;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.concurrent.TimeUnit;
import java.util.stream.Collectors;

/**
 * To run this code example, you need to create an AWS Identity and Access
 * Management (IAM) role with the correct policy as described in this
 * documentation:
 * https://docs.aws.amazon.com/pinpoint/latest/developerguide/audience-data-export.html
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ExportEndpoints {
    public static void main(String[] args) {
        final String usage = ""

        This program performs the following steps:

        1. Exports the endpoints to an Amazon S3 bucket.
        2. Downloads the exported endpoints files from Amazon S3.
    }
}
```

3. Parses the endpoints files to obtain the endpoint IDs and prints them.

```
Usage: ExportEndpoints <applicationId> <s3BucketName>
<iamExportRoleArn> <path>
```

Where:

applicationId - The ID of the Amazon Pinpoint application that has the endpoint.

s3BucketName - The name of the Amazon S3 bucket to export the JSON file to.\s

iamExportRoleArn - The ARN of an IAM role that grants Amazon Pinpoint write permissions to the S3 bucket. path - The path where the files downloaded from the Amazon S3 bucket are written (for example, C:/AWS/).

```
""";
```

```
if (args.length != 4) {
    System.out.println(usage);
    System.exit(1);
}

String applicationId = args[0];
String s3BucketName = args[1];
String iamExportRoleArn = args[2];
String path = args[3];
System.out.println("Deleting an application with ID: " + applicationId);

Region region = Region.US_EAST_1;
PinpointClient pinpoint = PinpointClient.builder()
    .region(region)
    .build();

S3Client s3Client = S3Client.builder()
    .region(region)
    .build();

exportAllEndpoints(pinpoint, s3Client, applicationId, s3BucketName, path,
iamExportRoleArn);
pinpoint.close();
s3Client.close();
}

public static void exportAllEndpoints(PinpointClient pinpoint,
    S3Client s3Client,
    String applicationId,
```

```

        String s3BucketName,
        String path,
        String iamExportRoleArn) {

    try {
        List<String> objectKeys = exportEndpointsToS3(pinpoint, s3Client,
s3BucketName, iamExportRoleArn,
            applicationId);
        List<String> endpointFileKeys = objectKeys.stream().filter(o ->
o.endsWith(".gz"))
            .collect(Collectors.toList());
        downloadFromS3(s3Client, path, s3BucketName, endpointFileKeys);

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<String> exportEndpointsToS3(PinpointClient pinpoint, S3Client
s3Client, String s3BucketName,
        String iamExportRoleArn, String applicationId) {

    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-
HH_mm:ss.SSS_z");
    String endpointsKeyPrefix = "exports/" + applicationId + "_" +
dateFormat.format(new Date());
    String s3UrlPrefix = "s3://" + s3BucketName + "/" + endpointsKeyPrefix +
"/";

    List<String> objectKeys = new ArrayList<>();
    String key;

    try {
        // Defines the export job that Amazon Pinpoint runs.
        ExportJobRequest jobRequest = ExportJobRequest.builder()
            .roleArn(iamExportRoleArn)
            .s3UrlPrefix(s3UrlPrefix)
            .build();

        CreateExportJobRequest exportJobRequest =
CreateExportJobRequest.builder()
            .applicationId(applicationId)
            .exportJobRequest(jobRequest)
            .build();

```

```
        System.out.format("Exporting endpoints from Amazon Pinpoint application
%s to Amazon S3 " +
            "bucket %s . . .\n", applicationId, s3BucketName);

        CreateExportJobResponse exportResult =
pinpoint.createExportJob(exportJobRequest);
        String jobId = exportResult.exportJobResponse().id();
        System.out.println(jobId);
        printExportJobStatus(pinpoint, applicationId, jobId);

        ListObjectsV2Request v2Request = ListObjectsV2Request.builder()
            .bucket(s3BucketName)
            .prefix(endpointsKeyPrefix)
            .build();

        // Create a list of object keys.
        ListObjectsV2Response v2Response = s3Client.listObjectsV2(v2Request);
        List<S3Object> objects = v2Response.contents();
        for (S3Object object : objects) {
            key = object.key();
            objectKeys.add(key);
        }

        return objectKeys;

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

private static void printExportJobStatus(PinpointClient pinpointClient,
    String applicationId,
    String jobId) {

    GetExportJobResponse getExportJobResult;
    String status;

    try {
        // Checks the job status until the job completes or fails.
        GetExportJobRequest exportJobRequest = GetExportJobRequest.builder()
            .jobId(jobId)
```

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

    do {
        getExportJobResult = pinpointClient.getExportJob(exportJobRequest);
        status =
getExportJobResult.exportJobResponse().jobStatus().toString().toUpperCase();
        System.out.format("Export job %s . . .\n", status);
        TimeUnit.SECONDS.sleep(3);

    } while (!status.equals("COMPLETED") && !status.equals("FAILED"));

    if (status.equals("COMPLETED")) {
        System.out.println("Finished exporting endpoints.");
    } else {
        System.err.println("Failed to export endpoints.");
        System.exit(1);
    }

} catch (PinpointException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Download files from an Amazon S3 bucket and write them to the path location.
public static void downloadFromS3(S3Client s3Client, String path, String
s3BucketName, List<String> objectKeys) {

    String newPath;
    try {
        for (String key : objectKeys) {
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                .bucket(s3BucketName)
                .key(key)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            String fileSuffix = new
SimpleDateFormat("yyyyMMddHHmmss").format(new Date());
```

```
        newPath = path + fileSuffix + ".gz";
        File myFile = new File(newPath);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
    }
    System.out.println("Download finished.");

} catch (S3Exception | NullPointerException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
}
```

- Untuk detail API, lihat [CreateExportJob](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan titik akhir

Contoh kode berikut menunjukkan cara mendapatkan titik akhir.

SDK for Java 2.x

#### Note

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

```
import com.google.gson.FieldNamingPolicy;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```



```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class LookUpEndpoint {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appId> <endpoint>

            Where:
                appId - The ID of the application to delete.
                endpoint - The ID of the endpoint.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpoint = args[1];
        System.out.println("Looking up an endpoint point with ID: " + endpoint);
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        lookupPinpointEndpoint(pinpoint, appId, endpoint);
        pinpoint.close();
    }

    public static void lookupPinpointEndpoint(PinpointClient pinpoint, String appId,
String endpoint) {
        try {
            GetEndpointRequest appRequest = GetEndpointRequest.builder()
                .applicationId(appId)
                .endpointId(endpoint)
                .build();

            GetEndpointResponse result = pinpoint.getEndpoint(appRequest);
            EndpointResponse endResponse = result.endpointResponse();

            // Uses the Google Gson library to pretty print the endpoint JSON.

```

```
Gson gson = new GsonBuilder()
    .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
    .setPrettyPrinting()
    .create();

String endpointJson = gson.toJson(endResponse);
System.out.println(endpointJson);

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.println("Done");
}
```

- Untuk detail API, lihat [GetEndpoint](#) di Referensi AWS SDK for Java 2.x API.

## Impor segmen

Contoh kode berikut menunjukkan cara mengimpor segmen.

## SDK for Java 2.x

### Note

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

## Impor segmen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.ImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.ImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.Format;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportSegment {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appId> <bucket> <key> <roleArn>\s

            Where:
                appId - The application ID to create a segment for.
                bucket - The name of the Amazon S3 bucket that contains the
segment definitons.
                key - The key of the S3 object.
                roleArn - ARN of the role that allows Amazon Pinpoint to
access S3. You need to set trust management for this to work. See https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\_policies\_elements\_principal.html
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String bucket = args[1];
        String key = args[2];
        String roleArn = args[3];

        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ImportJobResponse response = createImportSegment(pinpoint, appId, bucket,
key, roleArn);
        System.out.println("Import job for " + bucket + " submitted.");
        System.out.println("See application " + response.applicationId() + " for
import job status.");
    }
}
```

```
        System.out.println("See application " + response.jobStatus() + " for import
job status.");
        pinpoint.close();
    }

    public static ImportJobResponse createImportSegment(PinpointClient client,
        String appId,
        String bucket,
        String key,
        String roleArn) {

        try {
            ImportJobRequest importRequest = ImportJobRequest.builder()
                .defineSegment(true)
                .registerEndpoints(true)
                .roleArn(roleArn)
                .format(Format.JSON)
                .s3Url("s3://" + bucket + "/" + key)
                .build();

            CreateImportJobRequest jobRequest = CreateImportJobRequest.builder()
                .importJobRequest(importRequest)
                .applicationId(appId)
                .build();

            CreateImportJobResponse jobResponse =
client.createImportJob(jobRequest);
            return jobResponse.importJobResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- Untuk detail API, lihat [CreateImportJob](#) di Referensi AWS SDK for Java 2.x API.

## Daftar titik akhir

Contoh kode berikut menunjukkan bagaimana untuk daftar endpoint.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListEndpointIds {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <applicationId> <userId>

                Where:
                    applicationId - The ID of the Amazon Pinpoint application that
has the endpoint.
                    userId - The user id applicable to the endpoints""";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String applicationId = args[0];
        String userId = args[1];
```

```

        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllEndpoints(pinpoint, applicationId, userId);
        pinpoint.close();
    }

    public static void listAllEndpoints(PinpointClient pinpoint,
        String applicationId,
        String userId) {

        try {
            GetUserEndpointsRequest endpointsRequest =
                GetUserEndpointsRequest.builder()
                    .userId(userId)
                    .applicationId(applicationId)
                    .build();

            GetUserEndpointsResponse response =
                pinpoint.getUserEndpoints(endpointsRequest);
            List<EndpointResponse> endpoints = response.endpointsResponse().item();

            // Display the results.
            for (EndpointResponse endpoint : endpoints) {
                System.out.println("The channel type is: " +
                    endpoint.channelType());
                System.out.println("The address is " + endpoint.address());
            }

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Untuk detail API, lihat [GetUserEndpoints](#) di Referensi AWS SDK for Java 2.x API.

## Daftar segmen

Contoh kode berikut menunjukkan bagaimana untuk daftar segmen.

## SDK for Java 2.x

 Note

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

Daftar segmen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListSegments {
    public static void main(String[] args) {
        final String usage = ""

                Usage:  <appId>

                Where:
                    appId - The ID of the application that contains a segment.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
```

```
PinpointClient pinpoint = PinpointClient.builder()
    .region(Region.US_EAST_1)
    .build();

listSegs(pinpoint, appId);
pinpoint.close();
}

public static void listSegs(PinpointClient pinpoint, String appId) {
    try {
        GetSegmentsRequest request = GetSegmentsRequest.builder()
            .applicationId(appId)
            .build();

        GetSegmentsResponse response = pinpoint.getSegments(request);
        List<SegmentResponse> segments = response.segmentsResponse().item();
        for (SegmentResponse segment : segments) {
            System.out
                .println("Segement " + segment.id() + " " + segment.name() +
                    " " + segment.lastModifiedDate());
        }


    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [GetSegments](#) di Referensi AWS SDK for Java 2.x API.

Kirim email dan pesan teks

Contoh kode berikut menunjukkan cara mengirim email dan pesan teks dengan Amazon Pinpoint.

SDK for Java 2.x

 Note

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



## Kirim pesan email.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmailPart;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmail;
import software.amazon.awssdk.services.pinpoint.model.EmailMessage;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;

import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendEmailMessage {

    // The character encoding the you want to use for the subject line and
    // message body of the email.
    public static String charset = "UTF-8";

    // The body of the email for recipients whose email clients support HTML
    content.
    static final String body = ""
        Amazon Pinpoint test (AWS SDK for Java 2.x)
```

This email was sent through the Amazon Pinpoint Email API using the AWS SDK for Java 2.x

```
""";

public static void main(String[] args) {
    final String usage = ""

        Usage:    <subject> <appId> <senderAddress>
<toAddress>

    Where:
        subject - The email subject to use.
        senderAddress - The from address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
        toAddress - The to address. This address has to be verified in Amazon
Pinpoint in the region you're using to send email\s
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String subject = args[0];
    String senderAddress = args[1];
    String toAddress = args[2];
    System.out.println("Sending a message");
    PinpointEmailClient pinpoint = PinpointEmailClient.builder()
        .region(Region.US_EAST_1)
        .build();

    sendEmail(pinpoint, subject, senderAddress, toAddress);
    System.out.println("Email was sent");
    pinpoint.close();
}

public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress) {
    try {
        Content content = Content.builder()
            .data(body)
            .build();
```

```
        Body messageBody = Body.builder()
            .text(content)
            .build();

        Message message = Message.builder()
            .body(messageBody)
            .subject(Content.builder().data(subject).build())
            .build();

        Destination destination = Destination.builder()
            .toAddresses(toAddress)
            .build();

        EmailContent emailContent = EmailContent.builder()
            .simple(message)
            .build();

        SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(senderAddress)
            .destination(destination)
            .content(emailContent)
            .build();

        pinpointEmailClient.sendEmail(sendEmailRequest);
        System.out.println("Message Sent");

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Kirim pesan email dengan nilai CC.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
```

```
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendEmailMessageCC {

    // The body of the email.
    static final String body = ""
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS SDK
    for Java 2.x

    "";
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <subject> <senderAddress> <toAddress> <ccAddress>

            Where:
                subject - The email subject to use.
                senderAddress - The from address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
                toAddress - The to address. This address has to be verified in Amazon
Pinpoint in the region you're using to send email\s
                ccAddress - The CC address.

            "";

            if (args.length != 4) {
                System.out.println(usage);
                System.exit(1);
            }

            String subject = args[0];
            String senderAddress = args[1];
            String toAddress = args[2];
```

```
String ccAddress = args[3];

System.out.println("Sending a message");
PinpointEmailClient pinpoint = PinpointEmailClient.builder()
    .region(Region.US_EAST_1)
    .build();

ArrayList<String> ccList = new ArrayList<>();
ccList.add(ccAddress);
sendEmail(pinpoint, subject, senderAddress, toAddress, ccList);
pinpoint.close();
}

public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress, ArrayList<String> ccAddresses) {
    try {
        Content content = Content.builder()
            .data(body)
            .build();

        Body messageBody = Body.builder()
            .text(content)
            .build();

        Message message = Message.builder()
            .body(messageBody)
            .subject(Content.builder().data(subject).build())
            .build();

        Destination destination = Destination.builder()
            .toAddresses(toAddress)
            .ccAddresses(ccAddresses)
            .build();

        EmailContent emailContent = EmailContent.builder()
            .simple(message)
            .build();

        SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(senderAddress)
            .destination(destination)
            .content(emailContent)
            .build();
```

```
        pinpointEmailClient.sendEmail(sendEmailRequest);
        System.out.println("Message Sent");

    } catch (PinpointException e) {
        // Handle exception
        e.printStackTrace();
    }
}
}
```

## Kirim pesan SMS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessage {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
```

```

public static String registeredKeyword = "myKeyword";

// The sender ID to use when sending the message. Support for sender ID
// varies by country or region. For more information, see
// https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
public static String senderId = "MySenderId";

public static void main(String[] args) {
    final String usage = ""

        Usage:  <message> <appId> <originationNumber>
<destinationNumber>\s

        Where:
            message - The body of the message to send.
            appId - The Amazon Pinpoint project/application ID
to use when you send this message.
            originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
            destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s

        """;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String message = args[0];
    String appId = args[1];
    String originationNumber = args[2];
    String destinationNumber = args[3];
    System.out.println("Sending a message");
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber);
    pinpoint.close();
}

```

```
public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
    String originationNumber,
    String destinationNumber) {
    try {
        Map<String, AddressConfiguration> addressMap = new
HashMap<String, AddressConfiguration>();
        AddressConfiguration addConfig =
AddressConfiguration.builder()
            .channelType(ChannelType.SMS)
            .build();

        addressMap.put(destinationNumber, addConfig);
        SMSMessage smsMessage = SMSMessage.builder()
            .body(message)
            .messageType(messageType)
            .originationNumber(originationNumber)
            .senderId(senderId)
            .keyword(registeredKeyword)
            .build();

        // Create a DirectMessageConfiguration object.
        DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
            .smsMessage(smsMessage)
            .build();

        MessageRequest msgReq = MessageRequest.builder()
            .addresses(addressMap)
            .messageConfiguration(direct)
            .build();

        // create a SendMessagesRequest object
        SendMessagesRequest request = SendMessagesRequest.builder()
            .applicationId(appId)
            .messageRequest(msgReq)
            .build();

        SendMessagesResponse response =
pinpoint.sendMessages(request);
        MessageResponse msg1 = response.messageResponse();
        Map map1 = msg1.result();
    }
}
```



```

        // Write out the result of sendMessage.
        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

## Kirim pesan SMS batch.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageBatch {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.

```

```

public static String registeredKeyword = "myKeyword";

// The sender ID to use when sending the message. Support for sender ID
// varies by country or region. For more information, see
// https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
public static String senderId = "MySenderId";

public static void main(String[] args) {
    final String usage = ""

        Usage:  <message> <appId> <originationNumber>
<destinationNumber> <destinationNumber1>\s

        Where:
            message - The body of the message to send.
            appId - The Amazon Pinpoint project/application ID
to use when you send this message.
            originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
            destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).
            destinationNumber1 - The second recipient's phone
number. For best results, you should specify the phone number in E.164 format (for
example, +1-555-555-5654).\s

        """;

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String message = args[0];
    String appId = args[1];
    String originationNumber = args[2];
    String destinationNumber = args[3];
    String destinationNumber1 = args[4];
    System.out.println("Sending a message");
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

```

```

        sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber, destinationNumber1);
        pinpoint.close();
    }

    public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
        String originationNumber,
        String destinationNumber, String destinationNumber1) {
    try {
        Map<String, AddressConfiguration> addressMap = new
HashMap<String, AddressConfiguration>();
        AddressConfiguration addConfig =
AddressConfiguration.builder()
            .channelType(ChannelType.SMS)
            .build();

        // Add an entry to the Map object for each number to whom
you want to send a
        // message.
        addressMap.put(destinationNumber, addConfig);
        addressMap.put(destinationNumber1, addConfig);
        SMSMessage smsMessage = SMSMessage.builder()
            .body(message)
            .messageType(messageType)
            .originationNumber(originationNumber)
            .senderId(senderId)
            .keyword(registeredKeyword)
            .build();

        // Create a DirectMessageConfiguration object.
        DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
            .smsMessage(smsMessage)
            .build();

        MessageRequest msgReq = MessageRequest.builder()
            .addresses(addressMap)
            .messageConfiguration(direct)
            .build();

        // Create a SendMessagesRequest object.
        SendMessagesRequest request = SendMessagesRequest.builder()
            .applicationId(appId)

```

```

                .messageRequest(msgReq)
                .build();

        SendMessagesResponse response =
pinpoint.sendMessage(request);
        MessageResponse msg1 = response.messageResponse();
        Map map1 = msg1.result();

        // Write out the result of sendMessage.
        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [SendMessage](#) di Referensi AWS SDK for Java 2.x API.

Perbarui titik akhir

Contoh kode berikut menunjukkan cara memperbarui titik akhir.

SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.EndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

```

```
import software.amazon.awssdk.services.pinpoint.model.EndpointDemographic;
import software.amazon.awssdk.services.pinpoint.model.EndpointLocation;
import software.amazon.awssdk.services.pinpoint.model.EndpointUser;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.List;
import java.util.UUID;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Date;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateEndpoint {
    public static void main(String[] args) {
        final String usage = ""

            Usage: <appId>

            Where:
                appId - The ID of the application to create an endpoint for.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        EndpointResponse response = createEndpoint(pinpoint, appId);
        System.out.println("Got Endpoint: " + response.id());
        pinpoint.close();
    }
}
```

```
}

    public static EndpointResponse createEndpoint(PinpointClient client, String
appId) {
        String endpointId = UUID.randomUUID().toString();
        System.out.println("Endpoint ID: " + endpointId);

        try {
            EndpointRequest endpointRequest = createEndpointRequestData();
            UpdateEndpointRequest updateEndpointRequest =
UpdateEndpointRequest.builder()
                .applicationId(appId)
                .endpointId(endpointId)
                .endpointRequest(endpointRequest)
                .build();

            UpdateEndpointResponse updateEndpointResponse =
client.updateEndpoint(updateEndpointRequest);
            System.out.println("Update Endpoint Response: " +
updateEndpointResponse.messageBody());

            GetEndpointRequest getEndpointRequest = GetEndpointRequest.builder()
                .applicationId(appId)
                .endpointId(endpointId)
                .build();

            GetEndpointResponse getEndpointResponse =
client.getEndpoint(getEndpointRequest);
            System.out.println(getEndpointResponse.endpointResponse().address());

            System.out.println(getEndpointResponse.endpointResponse().channelType());

            System.out.println(getEndpointResponse.endpointResponse().applicationId());

            System.out.println(getEndpointResponse.endpointResponse().endpointStatus());
            System.out.println(getEndpointResponse.endpointResponse().requestId());
            System.out.println(getEndpointResponse.endpointResponse().user());

            return getEndpointResponse.endpointResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
        return null;
    }

    private static EndpointRequest createEndpointRequestData() {
        try {
            List<String> favoriteTeams = new ArrayList<>();
            favoriteTeams.add("Lakers");
            favoriteTeams.add("Warriors");
            HashMap<String, List<String>> customAttributes = new HashMap<>();
            customAttributes.put("team", favoriteTeams);

            EndpointDemographic demographic = EndpointDemographic.builder()
                .appVersion("1.0")
                .make("apple")
                .model("iPhone")
                .modelVersion("7")
                .platform("ios")
                .platformVersion("10.1.1")
                .timezone("America/Los_Angeles")
                .build();

            EndpointLocation location = EndpointLocation.builder()
                .city("Los Angeles")
                .country("US")
                .latitude(34.0)
                .longitude(-118.2)
                .postalCode("90068")
                .region("CA")
                .build();

            Map<String, Double> metrics = new HashMap<>();
            metrics.put("health", 100.00);
            metrics.put("luck", 75.00);

            EndpointUser user = EndpointUser.builder()
                .userId(UUID.randomUUID().toString())
                .build();

            DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm'Z'"); // Quoted
            // "Z" to indicate UTC, no timezone                                // offset

            String nowAsISO = df.format(new Date());

            return EndpointRequest.builder()
```

```

        .address(UUID.randomUUID().toString())
        .attributes(customAttributes)
        .channelType("APNS")
        .demographic(demographic)
        .effectiveDate(nowAsISO)
        .location(location)
        .metrics(metrics)
        .optOut("NONE")
        .requestId(UUID.randomUUID().toString())
        .user(user)
        .build();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
}

```

- Untuk detail API, lihat [UpdateEndpoint](#) di Referensi AWS SDK for Java 2.x API.

## Perbarui saluran

Contoh kode berikut menunjukkan cara memperbarui saluran.

### SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelResponse;
import software.amazon.awssdk.services.pinpoint.model.GetSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelRequest;

```



```
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateChannel {
    public static void main(String[] args) {
        final String usage = ""

            Usage: CreateChannel <appId>

            Where:
                appId - The name of the application whose channel is updated.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        SMSChannelResponse getResponse = getSmsChannel(pinpoint, appId);
        toggleSmsChannel(pinpoint, appId, getResponse);
        pinpoint.close();
    }

    private static SMSChannelResponse getSmsChannel(PinpointClient client, String
    appId) {
        try {
            GetSmsChannelRequest request = GetSmsChannelRequest.builder()
                .applicationId(appId)
                .build();
        }
    }
}
```

```
        SMSChannelResponse response =
client.getSmsChannel(request).smsChannelResponse();
        System.out.println("Channel state is " + response.enabled());
        return response;

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

private static void toggleSmsChannel(PinpointClient client, String appId,
SMSChannelResponse getResponse) {
    boolean enabled = !getResponse.enabled();
    try {
        SMSChannelRequest request = SMSChannelRequest.builder()
            .enabled(enabled)
            .build();

        UpdateSmsChannelRequest updateRequest =
UpdateSmsChannelRequest.builder()
            .smsChannelRequest(request)
            .applicationId(appId)
            .build();

        UpdateSmsChannelResponse result =
client.updateSmsChannel(updateRequest);
        System.out.println("Channel state: " +
result.smsChannelResponse().enabled());

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [GetSmsChannel](#) di Referensi AWS SDK for Java 2.x API.

## Amazon Pinpoint SMS dan Voice API contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum AWS SDK for Java 2.x dengan menggunakan Amazon Pinpoint SMS dan Voice API.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

### Tindakan

Kirim pesan suara dengan Amazon Pinpoint SMS dan Voice API

Contoh kode berikut menunjukkan cara mengirim pesan suara dengan Amazon Pinpoint SMS dan Voice API.

SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpointsmsvoice.PinpointSmsVoiceClient;
import software.amazon.awssdk.services.pinpointsmsvoice.model.SSMLMessageType;
import software.amazon.awssdk.services.pinpointsmsvoice.model.VoiceMessageContent;
```

```
import
    software.amazon.awssdk.services.pinpointsmsvoice.model.SendVoiceMessageRequest;
import
    software.amazon.awssdk.services.pinpointsmsvoice.model.PinpointSmsVoiceException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendVoiceMessage {

    // The Amazon Polly voice that you want to use to send the message. For a
    list
    // of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
    static final String voiceName = "Matthew";

    // The language to use when sending the message. For a list of supported
    // languages, see
    // https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
    static final String languageCode = "en-US";

    // The content of the message. This example uses SSML to customize and
    control
    // certain aspects of the message, such as by adding pauses and changing
    // phonation. The message can't contain any line breaks.
    static final String ssmlMessage = "<speak>This is a test message sent from "
        + "<emphasis>Amazon Pinpoint</emphasis> "
        + "using the <break strength='weak'/>AWS "
        + "SDK for Java. "
        + "<amazon:effect phonation='soft'>Thank "
        + "you for listening.</amazon:effect></speak>";

    public static void main(String[] args) {

        final String usage = ""
```

```
Usage: <originationNumber> <destinationNumber>\s
```

Where:

`originationNumber` - The phone number or short code that you specify has to be associated with your Amazon Pinpoint account. For best results, specify long codes in E.164 format (for example, +1-555-555-5654).

`destinationNumber` - The recipient's phone number. For best results, you should specify the phone number in E.164 format (for example, +1-555-555-5654).\s

```
""";
```

```
if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}
```

```
String originationNumber = args[0];
String destinationNumber = args[1];
System.out.println("Sending a voice message");
```

```
// Set the content type to application/json.
List<String> listVal = new ArrayList<>();
listVal.add("application/json");
Map<String, List<String>> values = new HashMap<>();
values.put("Content-Type", listVal);
```

```
ClientOverrideConfiguration config2 =
ClientOverrideConfiguration.builder()
    .headers(values)
    .build();
```

```
PinpointSmsVoiceClient client = PinpointSmsVoiceClient.builder()
    .overrideConfiguration(config2)
    .region(Region.US_EAST_1)
    .build();
```

```
sendVoiceMsg(client, originationNumber, destinationNumber);
client.close();
```

```
}
```

```
public static void sendVoiceMsg(PinpointSmsVoiceClient client, String
originationNumber,
    String destinationNumber) {
    try {
```

```
        SSMLMessageType ssmlMessageType = SSMLMessageType.builder()
            .languageCode(languageCode)
            .text(ssmlMessage)
            .voiceId(voiceName)
            .build();

        VoiceMessageContent content = VoiceMessageContent.builder()
            .ssmlMessage(ssmlMessageType)
            .build();

        SendVoiceMessageRequest voiceMessageRequest =
SendVoiceMessageRequest.builder()
            .destinationPhoneNumber(destinationNumber)
            .originationPhoneNumber(originationNumber)
            .content(content)
            .build();

        client.sendVoiceMessage(voiceMessageRequest);
        System.out.println("The message was sent successfully.");

    } catch (PinpointSmsVoiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [SendVoiceMessage](#) di Referensi AWS SDK for Java 2.x API.

## Contoh Amazon Polly menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan Amazon Polly. AWS SDK for Java 2.x

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

## Topik

- [Tindakan](#)

## Tindakan

Dapatkan suara yang tersedia untuk sintesis

Contoh kode berikut menunjukkan cara mendapatkan suara Amazon Polly yang tersedia untuk sintesis.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.Voice;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeVoicesSample {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
```

```
        .build();

        describeVoice(polly);
        polly.close();
    }

    public static void describeVoice(PollyClient polly) {
        try {
            DescribeVoicesRequest voicesRequest = DescribeVoicesRequest.builder()
                .languageCode("en-US")
                .build();

            DescribeVoicesResponse enUsVoicesResult =
polly.describeVoices(voicesRequest);
            List<Voice> voices = enUsVoicesResult.voices();
            for (Voice myVoice : voices) {
                System.out.println("The ID of the voice is " + myVoice.id());
                System.out.println("The gender of the voice is " +
myVoice.gender());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DescribeVoices](#) di Referensi AWS SDK for Java 2.x API.

## Daftar leksikon pengucapan

Contoh kode berikut menunjukkan cara mencantumkan leksikon pengucapan Amazon Polly.

## SDK for Java 2.x

### Note

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



```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.ListLexiconsResponse;
import software.amazon.awssdk.services.polly.model.ListLexiconsRequest;
import software.amazon.awssdk.services.polly.model.LexiconDescription;
import software.amazon.awssdk.services.polly.model.PollyException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListLexicons {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .build();

        listLexicons(polly);
        polly.close();
    }

    public static void listLexicons(PollyClient client) {
        try {
            ListLexiconsRequest listLexiconsRequest = ListLexiconsRequest.builder()
                .build();

            ListLexiconsResponse listLexiconsResult =
client.listLexicons(listLexiconsRequest);
            List<LexiconDescription> lexiconDescription =
listLexiconsResult.lexicons();
            for (LexiconDescription lexDescription : lexiconDescription) {
                System.out.println("The name of the Lexicon is " +
lexDescription.name());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

```
    }  
  }  
}
```

- Untuk detail API, lihat [ListLexicons](#) di Referensi AWS SDK for Java 2.x API.

## Sintesis ucapan dari teks

Contoh kode berikut menunjukkan cara mensintesis ucapan dari teks dengan Amazon Polly.

## SDK for Java 2.x

### Note

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

```
import javazoom.jl.decoder.JavaLayerException;  
import software.amazon.awssdk.core.ResponseInputStream;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.polly.PollyClient;  
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;  
import software.amazon.awssdk.services.polly.model.Voice;  
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;  
import software.amazon.awssdk.services.polly.model.OutputFormat;  
import software.amazon.awssdk.services.polly.model.PollyException;  
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechRequest;  
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechResponse;  
import java.io.IOException;  
import java.io.InputStream;  
import javazoom.jl.player.advanced.AdvancedPlayer;  
import javazoom.jl.player.advanced.PlaybackEvent;  
import javazoom.jl.player.advanced.PlaybackListener;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class PollyDemo {
    private static final String SAMPLE = "Congratulations. You have successfully
built this working demo " +
        " of Amazon Polly in Java Version 2. Have fun building voice enabled
apps with Amazon Polly (that's me!), and always "
    +
        " look at the AWS website for tips and tricks on using Amazon Polly and
other great services from AWS";

    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .build();

        talkPolly(polly);
        polly.close();
    }

    public static void talkPolly(PollyClient polly) {
        try {
            DescribeVoicesRequest describeVoiceRequest =
DescribeVoicesRequest.builder()
                .engine("standard")
                .build();

            DescribeVoicesResponse describeVoicesResult =
polly.describeVoices(describeVoiceRequest);
            Voice voice = describeVoicesResult.voices().stream()
                .filter(v -> v.name().equals("Joanna"))
                .findFirst()
                .orElseThrow(() -> new RuntimeException("Voice not found"));
            InputStream stream = synthesize(polly, SAMPLE, voice, OutputFormat.MP3);
            AdvancedPlayer player = new AdvancedPlayer(stream,

javazoom.jl.player.FactoryRegistry.systemRegistry().createAudioDevice());
            player.setPlaybackListener(new PlaybackListener() {
                public void playbackStarted(PlaybackEvent evt) {
                    System.out.println("Playback started");
                    System.out.println(SAMPLE);
                }
            });

            public void playbackFinished(PlaybackEvent evt) {
```

```
        System.out.println("Playback finished");
    }
});

// play it!
player.play();

} catch (PollyException | JavaLayerException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static InputStream synthesize(PollyClient polly, String text, Voice
voice, OutputFormat format)
    throws IOException {
    SynthesizeSpeechRequest synthReq = SynthesizeSpeechRequest.builder()
        .text(text)
        .voiceId(voice.id())
        .outputFormat(format)
        .build();

    ResponseInputStream<SynthesizeSpeechResponse> synthRes =
polly.synthesizeSpeech(synthReq);
    return synthRes;
}
}
```

- Untuk detail API, lihat [SynthesizeSpeech](#) di Referensi AWS SDK for Java 2.x API.

## Contoh Amazon RDS menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan RDS AWS SDK for Java 2.x with Amazon.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

## Memulai

### Halo Amazon RDS

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon RDS.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }
}
```

```
}

public static void describeInstances(RdsClient rdsClient) {
    try {
        DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
        List<DBInstance> instanceList = response.dbInstances();
        for (DBInstance instance : instanceList) {
            System.out.println("Instance ARN is: " + instance.dbInstanceArn());
            System.out.println("The Engine is " + instance.engine());
            System.out.println("Connection endpoint is" +
instance.endpoint().address());
        }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Lihat detail API di [DescribeDBInstances](#) dalam Referensi API AWS SDK for Java 2.x .

## Topik

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

## Tindakan

Buat instans basis data

Contoh kode berikut menunjukkan cara membuat instans Amazon RDS DB dan menunggu sampai tersedia.

SDK for Java 2.x

### Note

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

```
import com.google.gson.Gson;
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For more details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating\_how-services-use-secrets\_RS.html
 */

public class CreateDBInstance {
    public static long sleepTime = 20;

    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <dbInstanceIdentifier> <dbName> <secretName>
```

```

        Where:
            dbInstanceIdentifier - The database instance identifier.\s
            dbName - The database name.\s
            secretName - The name of the AWS Secrets Manager secret that
contains the database credentials."
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String dbInstanceIdentifier = args[0];
    String dbName = args[1];
    String secretName = args[2];
    Gson gson = new Gson();
    User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
    Region region = Region.US_WEST_2;
    RdsClient rdsClient = RdsClient.builder()
        .region(region)
        .build();

    createDatabaseInstance(rdsClient, dbInstanceIdentifier, dbName,
user.getUsername(), user.getPassword());
    waitForInstanceReady(rdsClient, dbInstanceIdentifier);
    rdsClient.close();
}

private static SecretsManagerClient getSecretClient() {
    Region region = Region.US_WEST_2;
    return SecretsManagerClient.builder()
        .region(region)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();
}

private static String getSecretValues(String secretName) {
    SecretsManagerClient secretClient = getSecretClient();
    GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
        .secretId(secretName)
        .build();
}

```



```
        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void createDatabaseInstance(RdsClient rdsClient,
        String dbInstanceIdentifier,
        String dbName,
        String userName,
        String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .allocatedStorage(100)
                .dbName(dbName)
                .engine("mysql")
                .dbInstanceClass("db.m4.large")
                .engineVersion("8.0")
                .storageType("standard")
                .masterUsername(userName)
                .masterUserPassword(userPassword)
                .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
```

```
DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
    .dbInstanceIdentifier(dbInstanceIdentifier)
    .build();

// Loop until the cluster is ready.
while (!instanceReady) {
    DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
    List<DBInstance> instanceList = response.dbInstances();
    for (DBInstance instance : instanceList) {
        instanceReadyStr = instance.dbInstanceStatus();
        if (instanceReadyStr.contains("available"))
            instanceReady = true;
        else {
            System.out.print(".");
            Thread.sleep(sleepTime * 1000);
        }
    }
}
System.out.println("Database instance is available!");


} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Lihat detail API di [CreateDBInstance](#) dalam Referensi API AWS SDK for Java 2.x .

Buat grup parameter basis data

Contoh kode berikut menunjukkan cara membuat grup parameter Amazon RDS DB.

SDK for Java 2.x

 Note

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

```
public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
        .dbParameterGroupName(dbGroupName)
        .dbParameterGroupFamily(dbParameterGroupFamily)
        .description("Created by using the AWS SDK for Java")
        .build();

        CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());


    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [CreateDB ParameterGroup](#) di AWS SDK for Java 2.x Referensi API.

Buat cuplikan instans basis data

Contoh kode berikut menunjukkan cara membuat snapshot dari instans Amazon RDS DB.

SDK for Java 2.x

 Note

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

```
// Create an Amazon RDS snapshot.
public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
    try {
```

```

        CreateDbSnapshotRequest snapshotRequest =
        CreateDbSnapshotRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbSnapshotResponse response =
        rdsClient.createDBSnapshot(snapshotRequest);
        System.out.println("The Snapshot id is " +
        response.dbSnapshot().dbiResourceId());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

```

- Lihat detail API di [CreateDBSnapshot](#) dalam Referensi API AWS SDK for Java 2.x .

## Buat token autentikasi

Contoh kode berikut menunjukkan cara membuat token autentikasi untuk autentikasi IAM.

## SDK for Java 2.x

### Note

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

Gunakan [RdsUtilities](#) kelas untuk menghasilkan token otentikasi.

```

public class GenerateRDSAuthToken {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <dbInstanceIdentifier> <masterUsername>

                Where:

```

```
        dbInstanceIdentifier - The database instance identifier.\s
        masterUsername - The master user name.\s
        """";

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String dbInstanceIdentifier = args[0];
    String masterUsername = args[1];
    Region region = Region.US_WEST_2;
    RdsClient rdsClient = RdsClient.builder()
        .region(region)
        .build();

    String token = getAuthToken(rdsClient, dbInstanceIdentifier,
masterUsername);
    System.out.println("The token response is " + token);
}

    public static String getAuthToken(RdsClient rdsClient, String
dbInstanceIdentifier, String masterUsername) {

        RdsUtilities utilities = rdsClient.utilities();
        try {
            GenerateAuthenticationTokenRequest tokenRequest =
GenerateAuthenticationTokenRequest.builder()
                .credentialsProvider(ProfileCredentialsProvider.create())
                .username(masterUsername)
                .port(3306)
                .hostname(dbInstanceIdentifier)
                .build();

            return utilities.generateAuthenticationToken(tokenRequest);

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Untuk detail API, lihat [GeneraTerds AuthToken di Referensi AWS SDK for Java 2.x API](#).

## Hapus instans basis data

Contoh kode berikut menunjukkan cara menghapus instans Amazon RDS DB.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDBInstance {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <dbInstanceIdentifier>\s

                Where:
                dbInstanceIdentifier - The database instance identifier\s
        """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .deleteAutomatedBackups(true)
                .skipFinalSnapshot(true)
                .build();

            DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());


        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Lihat detail API di [DeleteDBInstance](#) dalam Referensi API AWS SDK for Java 2.x .

## Hapus grup parameter basis data

Contoh kode berikut menunjukkan cara menghapus grup parameter Amazon RDS DB.

## SDK for Java 2.x

 Note

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

```
// Delete the parameter group after database has been deleted.
// An exception is thrown if you attempt to delete the para group while database
// exists.
public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
    throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(dbARN) == 0) {
                    System.out.println(dbARN + " still exists");
                    didFind = true;
                }
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.

                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }
}
```



```
        // Delete the para group.
        DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
        .dbParameterGroupName(dbGroupName)
        .build();

        rdsClient.deleteDBParameterGroup(parameterGroupRequest);
        System.out.println(dbGroupName + " was deleted.");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DeleteDB ParameterGroup](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan instans basis data

Contoh kode berikut menunjukkan cara mendeskripsikan instans Amazon RDS DB.

SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " + instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Lihat detail API di [DescribeDBInstances](#) dalam Referensi API AWS SDK for Java 2.x .

Jelaskan grup parameter basis data

Contoh kode berikut menunjukkan cara mendeskripsikan kelompok parameter Amazon RDS DB.

## SDK for Java 2.x

 Note

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

```
public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
    try {
        DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .maxRecords(20)
            .build();

        DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
        List<DBParameterGroup> groups = response.dbParameterGroups();
        for (DBParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbParameterGroupName());
            System.out.println("The group description is " +
group.description());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DescribeDB ParameterGroups](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan versi mesin basis data

Contoh kode berikut menunjukkan cara mendeskripsikan versi mesin database Amazon RDS.

## SDK for Java 2.x

 Note

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

```
public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .defaultOnly(true)
            .engine("mysql")
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engineOb : engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "
                + engineOb.dbParameterGroupFamily());
            System.out.println("The name of the database engine " +
engineOb.engine());
            System.out.println("The version number of the database engine " +
engineOb.engineVersion());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DescribeDB EngineVersions](#) di Referensi AWS SDK for Java 2.x API.

## Jelaskan opsi untuk instans basis data

Contoh kode berikut menunjukkan cara mendeskripsikan opsi untuk instans Amazon RDS DB.

SDK for Java 2.x

### Note

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

```
// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .engine("mysql")
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
        List<DBEngineVersion> dbEngines = response.dbEngineVersions();
        for (DBEngineVersion dbEngine : dbEngines) {
            System.out.println("The engine version is " +
dbEngine.engineVersion());
            System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DescribeOrderableDB InstanceOptions](#) di Referensi AWS SDK for Java 2.x API.

## Jelaskan parameter dalam grup parameter basis data

Contoh kode berikut menunjukkan cara mendeskripsikan parameter dalam grup parameter Amazon RDS DB.

SDK for Java 2.x

### Note

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

```
// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
    try {
        DescribeDbParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .source("user")
                .build();
        }

        DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
            if ((paraName.compareTo("auto_increment_offset") == 0)
                || (paraName.compareTo("auto_increment_increment ") == 0)) {
                System.out.println("*** The parameter name is " + paraName);
                System.out.println("*** The parameter value is " +
para.parameterValue());
            }
        }
    }
}
```

```
        System.out.println("*** The parameter data type is " +
para.dataType());
        System.out.println("*** The parameter description is " +
para.description());
        System.out.println("*** The parameter allowed values is " +
para.allowedValues());
    }
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Lihat detail API di [DescribeDBParameters](#) dalam Referensi API AWS SDK for Java 2.x .

## Ubah instans basis data

Contoh kode berikut menunjukkan cara memodifikasi instans Amazon RDS DB.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```

```

* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ModifyDBInstance {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbInstanceIdentifier> <dbSnapshotIdentifier>\s
            Where:
                dbInstanceIdentifier - The database instance identifier.\s
                masterUserPassword - The updated password that corresponds to
the master user name.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUserPassword = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        updateIntance(rdsClient, dbInstanceIdentifier, masterUserPassword);
        rdsClient.close();
    }

    public static void updateIntance(RdsClient rdsClient, String
dbInstanceIdentifier, String masterUserPassword) {
        try {
            // For a demo - modify the DB instance by modifying the master password.
            ModifyDbInstanceRequest modifyDbInstanceRequest =
ModifyDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .publiclyAccessible(true)
                .masterUserPassword(masterUserPassword)
                .build();

            ModifyDbInstanceResponse instanceResponse =
rdsClient.modifyDBInstance(modifyDbInstanceRequest);

```



```
        System.out.print("The ARN of the modified database is: " +
instanceResponse.dbInstance().dbInstanceArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Lihat detail API di [ModifyDBInstance](#) dalam Referensi API AWS SDK for Java 2.x .

But ulang instans basis data

Contoh kode berikut ini menunjukkan cara membut ulang instans basis data Amazon RDS.

SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RebootDBInstance {
    public static void main(String[] args) {
        final String usage = ""
```

```
Usage:
    <dbInstanceIdentifier>\s

Where:
    dbInstanceIdentifier - The database instance identifier\s
    """";
```

```
if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String dbInstanceIdentifier = args[0];
Region region = Region.US_WEST_2;
RdsClient rdsClient = RdsClient.builder()
    .region(region)
    .build();

rebootInstance(rdsClient, dbInstanceIdentifier);
rdsClient.close();
}

public static void rebootInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        RebootDbInstanceRequest rebootDbInstanceRequest =
RebootDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        RebootDbInstanceResponse instanceResponse =
rdsClient.rebootDBInstance(rebootDbInstanceRequest);
        System.out.print("The database " +
instanceResponse.dbInstance().dbInstanceArn() + " was rebooted");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
```

- Lihat detail API di [RebootDBInstance](#) dalam Referensi API AWS SDK for Java 2.x .

## Mengambil atribut-atribut

Contoh kode berikut menunjukkan cara mengambil atribut milik akun Amazon RDS.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.AccountQuota;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeAccountAttributesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAccountAttributes {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        getAccountAttributes(rdsClient);
        rdsClient.close();
    }

    public static void getAccountAttributes(RdsClient rdsClient) {
        try {
```

```

        DescribeAccountAttributesResponse response =
rdsClient.describeAccountAttributes();
        List<AccountQuota> quotasList = response.accountQuotas();
        for (AccountQuota quotas : quotasList) {
            System.out.println("Name is: " + quotas.accountQuotaName());
            System.out.println("Max value is " + quotas.max());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}

```

- Untuk detail API, lihat [DescribeAccountAttributes](#) di Referensi AWS SDK for Java 2.x API.

Perbarui parameter dalam grup parameter basis data

Contoh kode berikut menunjukkan cara memperbarui parameter dalam grup parameter Amazon RDS DB.

SDK for Java 2.x

#### Note

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

```

// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();

        List<Parameter> paraList = new ArrayList<>();
    }
}

```

```
        paraList.add(parameter1);
        ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
        .dbParameterGroupName(dbGroupName)
        .parameters(paraList)
        .build();

        ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
        System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ModifyDB ParameterGroup](#) di AWS SDK for Java 2.x Referensi API.

## Skenario

### Memulai instans basis data

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- 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 for Java 2.x

#### Note

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

## Menjalankan beberapa operasi.

```
import com.google.gson.Gson;
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotRequest;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotResponse;
import software.amazon.awssdk.services.rds.model.DBEngineVersion;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.DBParameterGroup;
import software.amazon.awssdk.services.rds.model.DBSnapshot;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsResponse;
import
    software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsResponse;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.OrderableDBInstanceOption;
import software.amazon.awssdk.services.rds.model.Parameter;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupRequest;
import
    software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbParameterGroupRequest;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import java.util.ArrayList;
import java.util.List;
```

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating\_how-services-use-secrets\_RS.html
 *
 * This Java example performs these tasks:
 *
 * 1. Returns a list of the available DB engines.
 * 2. Selects an engine family and create a custom DB parameter group.
 * 3. Gets the parameter groups.
 * 4. Gets parameters in the group.
 * 5. Modifies the auto_increment_offset parameter.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions.
 * 8. Gets a list of micro instance classes available for the selected engine.
 * 9. Creates an RDS database instance that contains a MySQL database and uses
 * the parameter group.
 * 10. Waits for the DB instance to be ready and prints out the connection
 * endpoint value.
 * 11. Creates a snapshot of the DB instance.
 * 12. Waits for an RDS DB snapshot to be ready.
 * 13. Deletes the RDS DB instance.
 * 14. Deletes the parameter group.
 */
public class RDSScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

        Usage:
```

```

        <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
<dbName> <dbSnapshotIdentifier> <secretName>

```

Where:

```

        dbGroupName - The database group name.\s
        dbParameterGroupFamily - The database parameter group name (for
example, mysql8.0).
        dbInstanceIdentifier - The database instance identifier\s
        dbName - The database name.\s
        dbSnapshotIdentifier - The snapshot identifier.\s
        secretName - The name of the AWS Secrets Manager secret that
contains the database credentials"

```

```

        """;

    if (args.length != 6) {
        System.out.println(usage);
        System.exit(1);
    }

    String dbGroupName = args[0];
    String dbParameterGroupFamily = args[1];
    String dbInstanceIdentifier = args[2];
    String dbName = args[3];
    String dbSnapshotIdentifier = args[4];
    String secretName = args[5];

    Gson gson = new Gson();
    User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
    String masterUsername = user.getUsername();
    String masterUserPassword = user.getPassword();

    Region region = Region.US_WEST_2;
    RdsClient rdsClient = RdsClient.builder()
        .region(region)
        .build();
    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon RDS example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Return a list of the available DB engines");
    describeDBEngines(rdsClient);
    System.out.println(DASHES);

```



```
System.out.println(DASHES);
System.out.println("2. Create a custom parameter group");
createDBParameterGroup(rdsClient, dbGroupName, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get the parameter group");
describeDbParameterGroups(rdsClient, dbGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get the parameters in the group");
describeDbParameters(rdsClient, dbGroupName, 0);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Modify the auto_increment_offset parameter");
modifyDBParas(rdsClient, dbGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Display the updated value");
describeDbParameters(rdsClient, dbGroupName, -1);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of allowed engine versions");
getAllowedEngines(rdsClient, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Get a list of micro instance classes available for
the selected engine");
getMicroInstances(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(
    "9. Create an RDS database instance that contains a MySQL database
and uses the parameter group");
String dbARN = createDatabaseInstance(rdsClient, dbGroupName,
dbInstanceIdentifier, dbName, masterUsername,
    masterUserPassword);
```

```
System.out.println("The ARN of the new database is " + dbARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Wait for DB instance to be ready");
waitForInstanceReady(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Create a snapshot of the DB instance");
createSnapshot(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Wait for DB snapshot to be ready");
waitForSnapshotReady(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Delete the DB instance");
deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Delete the parameter group");
deleteParaGroup(rdsClient, dbGroupName, dbARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The Scenario has successfully completed.");
System.out.println(DASHES);

rdsClient.close();
}

private static SecretsManagerClient getSecretClient() {
    Region region = Region.US_WEST_2;
    return SecretsManagerClient.builder()
        .region(region)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();
}
```

```

public static String getSecretValues(String secretName) {
    SecretsManagerClient secretClient = getSecretClient();
    GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
        .secretId(secretName)
        .build();

    GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
    return valueResponse.secretString();
}

// Delete the parameter group after database has been deleted.
// An exception is thrown if you attempt to delete the para group while database
// exists.
public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
    throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            didFind = false;
            int index = 1;
            for (DBInstance instance : instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(dbARN) == 0) {
                    System.out.println(dbARN + " still exists");
                    didFind = true;
                }
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.

                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }
}

```

```
    }

    // Delete the para group.
    DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
    .dbParameterGroupName(dbGroupName)
    .build();

    rdsClient.deleteDBParameterGroup(parameterGroupRequest);
    System.out.println(dbGroupName + " was deleted.");

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

// Delete the DB instance.
public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .deleteAutomatedBackups(true)
        .skipFinalSnapshot(true)
        .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the snapshot instance is available.
public static void waitForSnapshotReady(RdsClient rdsClient, String
dbInstanceIdentifier,
    String dbSnapshotIdentifier) {
    try {
```

```
        boolean snapshotReady = false;
        String snapshotReadyStr;
        System.out.println("Waiting for the snapshot to become available.");

        DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        while (!snapshotReady) {
            DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
            List<DBSnapshot> snapshotList = response.dbSnapshots();
            for (DBSnapshot snapshot : snapshotList) {
                snapshotReadyStr = snapshot.status();
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }

        System.out.println("The Snapshot is available!");
    } catch (RdsException | InterruptedException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Create an Amazon RDS snapshot.
public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
    try {
        CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
```

```
        System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        String endpoint = "";
        while (!instanceReady) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                instanceReadyStr = instance.dbInstanceStatus();
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint().address();
                    instanceReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
        System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

    } catch (RdsException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}

// Create a database instance and return the ARN of the database.
public static String createDatabaseInstance(RdsClient rdsClient,
    String dbGroupName,
    String dbInstanceIdentifier,
    String dbName,
    String masterUsername,
    String masterUserPassword) {

    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .allocatedStorage(100)
            .dbName(dbName)
            .dbParameterGroupName(dbGroupName)
            .engine("mysql")
            .dbInstanceClass("db.m4.large")
            .engineVersion("8.0")
            .storageType("standard")
            .masterUsername(masterUsername)
            .masterUserPassword(masterUserPassword)
            .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }

    return "";
}

// Get a list of micro instances.
public static void getMicroInstances(RdsClient rdsClient) {
    try {
        DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest
```

```
        .builder()
        .engine("mysql")
        .build();

DescribeOrderableDbInstanceOptionsResponse response = rdsClient
    .describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
List<OrderableDBInstanceOption> orderableDBInstances =
response.orderableDBInstanceOptions();
for (OrderableDBInstanceOption dbInstanceOption : orderableDBInstances)
{
    System.out.println("The engine version is " +
dbInstanceOption.engineVersion());
    System.out.println("The engine description is " +
dbInstanceOption.engine());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .engine("mysql")
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
        List<DBEngineVersion> dbEngines = response.dbEngineVersions();
        for (DBEngineVersion dbEngine : dbEngines) {
            System.out.println("The engine version is " +
dbEngine.engineVersion());
            System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
    }
}
```



```
        System.exit(1);
    }
}

// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();

        List<Parameter> paraList = new ArrayList<>();
        paraList.add(parameter1);
        ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .parameters(paraList)
            .build();

        ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
        System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
    try {
        DescribeDbParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)

```

```

        .source("user")
        .build();
    }

    DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
    List<Parameter> dbParameters = response.parameters();
    String paraName;
    for (Parameter para : dbParameters) {
        // Only print out information about either auto_increment_offset or
        // auto_increment_increment.
        paraName = para.parameterName();
        if ((paraName.compareTo("auto_increment_offset") == 0)
            || (paraName.compareTo("auto_increment_increment ") == 0)) {
            System.out.println("*** The parameter name is " + paraName);
            System.out.println("*** The parameter value is " +
para.parameterValue());
            System.out.println("*** The parameter data type is " +
para.dataType());
            System.out.println("*** The parameter description is " +
para.description());
            System.out.println("*** The parameter allowed values is " +
para.allowedValues());
        }
    }

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
    try {
        DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .maxRecords(20)
            .build();

        DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
        List<DBParameterGroup> groups = response.dbParameterGroups();

```

```
        for (DBParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbParameterGroupName());
            System.out.println("The group description is " +
group.description());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDBEngines(RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .defaultOnly(true)
            .engine("mysql")
            .maxRecords(20)
            .build();
```

```
DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
List<DBEngineVersion> engines = response.dbEngineVersions();

// Get all DBEngineVersion objects.
for (DBEngineVersion engineOb : engines) {
    System.out.println("The name of the DB parameter group family for
the database engine is "
        + engineOb.dbParameterGroupFamily());
    System.out.println("The name of the database engine " +
engineOb.engine());
    System.out.println("The version number of the database engine " +
engineOb.engineVersion());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [CreateDBInstance](#)
  - [dibuatB ParameterGroup](#)
  - [CreateDBSnapshot](#)
  - [DeleteDBInstance](#)
  - [DihapusB ParameterGroup](#)
  - [DijelaskanB EngineVersions](#)
  - [DescribeDBInstances](#)
  - [DijelaskanB ParameterGroups](#)
  - [DescribeDBParameters](#)
  - [DescribeDBSnapshots](#)
  - [DescribeOrderableDB InstanceOptions](#)
  - [ModifyDB ParameterGroup](#)

## Contoh Amazon Redshift menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with Amazon Redshift.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

### Tindakan

Membuat klaster

Contoh kode berikut menunjukkan cara membuat cluster Amazon Redshift.

SDK for Java 2.x

#### Note

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

Buat cluster.

```
public static void createCluster(RedshiftClient redshiftClient, String
clusterId, String masterUsername,
    String masterUserPassword) {
    try {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(masterUsername) // set the user name here
```

```
        .masterUserPassword(masterUserPassword) // set the user password
here
        .nodeType("dc2.large")
        .publiclyAccessible(true)
        .numberOfNodes(2)
        .build();

        CreateClusterResponse clusterResponse =
redshiftClient.createCluster(clusterRequest);
        System.out.println("Created cluster " +
clusterResponse.cluster().clusterIdentifier());

    } catch (RedshiftException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [CreateCluster](#) di Referensi AWS SDK for Java 2.x API.

## Hapus klaster

Contoh kode berikut menunjukkan cara menghapus cluster Amazon Redshift.

### SDK for Java 2.x

#### Note

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

## Hapus klaster .

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import software.amazon.awssdk.services.redshift.model.DeleteClusterRequest;
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteCluster {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <clusterId>\s

            Where:
                clusterId - The id of the cluster to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterId = args[0];
        Region region = Region.US_WEST_2;
        RedshiftClient redshiftClient = RedshiftClient.builder()
            .region(region)
            .build();

        deleteRedshiftCluster(redshiftClient, clusterId);
        redshiftClient.close();
    }

    public static void deleteRedshiftCluster(RedshiftClient redshiftClient, String
clusterId) {
        try {
            DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
                .clusterIdentifier(clusterId)
                .skipFinalClusterSnapshot(true)
                .build();

```

```
        DeleteClusterResponse response =
redshiftClient.deleteCluster(deleteClusterRequest);
        System.out.println("The status is " +
response.cluster().clusterStatus());

        } catch (RedshiftException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DeleteCluster](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan cluster Anda

Contoh kode berikut menunjukkan cara mendeskripsikan cluster Amazon Redshift Anda.

SDK for Java 2.x

#### Note

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

Jelaskan cluster.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import software.amazon.awssdk.services.redshift.model.Cluster;
import software.amazon.awssdk.services.redshift.model.DescribeClustersResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```



```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DescribeClusters {

    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        RedshiftClient redshiftClient = RedshiftClient.builder()
            .region(region)
            .build();

        describeRedshiftClusters(redshiftClient);
        redshiftClient.close();
    }

    public static void describeRedshiftClusters(RedshiftClient redshiftClient) {
        try {
            DescribeClustersResponse clusterResponse =
redshiftClient.describeClusters();
            List<Cluster> clusterList = clusterResponse.clusters();
            for (Cluster cluster : clusterList) {
                System.out.println("Cluster database name is: " + cluster.dbName());
                System.out.println("Cluster status is: " + cluster.clusterStatus());
            }

        } catch (RedshiftException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DescribeClusters](#) di Referensi AWS SDK for Java 2.x API.

## Memodifikasi klaster

Contoh kode berikut menunjukkan cara memodifikasi cluster Amazon Redshift.

## SDK for Java 2.x

 Note

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

## Memodifikasi cluster.

```
public static void modifyCluster(RedshiftClient redshiftClient, String
clusterId) {

    try {
        ModifyClusterRequest modifyClusterRequest =
ModifyClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .preferredMaintenanceWindow("wed:07:30-wed:08:00")
            .build();

        ModifyClusterResponse clusterResponse =
redshiftClient.modifyCluster(modifyClusterRequest);
        System.out.println("The modified cluster was successfully modified and
has "
            + clusterResponse.cluster().preferredMaintenanceWindow() + " as
the maintenance window");

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ModifyCluster](#) di Referensi AWS SDK for Java 2.x API.

## Contoh Amazon Rekognition menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x With Amazon Rekognition.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

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

## Tindakan

Bandungkan wajah dalam gambar dengan gambar referensi

Contoh kode berikut menunjukkan cara membandingkan wajah dalam gambar dengan gambar referensi dengan Amazon Rekognition.

Untuk informasi selengkapnya, lihat [Membandingkan wajah dalam gambar](#).

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.CompareFacesRequest;
import software.amazon.awssdk.services.rekognition.model.CompareFacesResponse;
import software.amazon.awssdk.services.rekognition.model.CompareFacesMatch;
import software.amazon.awssdk.services.rekognition.model.ComparedFace;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
```

```
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CompareFaces {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <pathSource> <pathTarget>

            Where:
                pathSource - The path to the source image (for example, C:\\AWS\\
\\pic1.png).\\s
                pathTarget - The path to the target image (for example, C:\\AWS\\
\\pic2.png).\\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        Float similarityThreshold = 70F;
        String sourceImage = args[0];
        String targetImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        compareTwoFaces(rekClient, similarityThreshold, sourceImage, targetImage);
        rekClient.close();
    }
}
```

```
public static void compareTwoFaces(RecognitionClient rekClient, Float
similarityThreshold, String sourceImage,
String targetImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        InputStream tarStream = new FileInputStream(targetImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        SdkBytes targetBytes = SdkBytes.fromInputStream(tarStream);

        // Create an Image object for the source image.
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        Image tarImage = Image.builder()
            .bytes(targetBytes)
            .build();

        CompareFacesRequest facesRequest = CompareFacesRequest.builder()
            .sourceImage(souImage)
            .targetImage(tarImage)
            .similarityThreshold(similarityThreshold)
            .build();

        // Compare the two images.
        CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
        List<CompareFacesMatch> faceDetails = compareFacesResult.faceMatches();
        for (CompareFacesMatch match : faceDetails) {
            ComparedFace face = match.face();
            BoundingBox position = face.boundingBox();
            System.out.println("Face at " + position.left().toString()
                + " " + position.top()
                + " matches with " + face.confidence().toString()
                + "% confidence.");
        }
        List<ComparedFace> uncompered = compareFacesResult.unmatchedFaces();
        System.out.println("There was " + uncompered.size() + " face(s) that did
not match");
        System.out.println("Source image rotation: " +
compareFacesResult.sourceImageOrientationCorrection());
        System.out.println("target image rotation: " +
compareFacesResult.targetImageOrientationCorrection());
    }
}
```

```
        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println("Failed to load source image " + sourceImage);
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [CompareFaces](#) di Referensi AWS SDK for Java 2.x API.

## Buat koleksi

Contoh kode berikut menunjukkan cara membuat koleksi Amazon Rekognition.

Untuk informasi selengkapnya, lihat [Membuat koleksi](#).

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCollection {
    public static void main(String[] args) {
        final String usage = ""
```

```
Usage:    <collectionName>\s

Where:
    collectionName - The name of the collection.\s
    """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String collectionId = args[0];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

System.out.println("Creating collection: " + collectionId);
createMyCollection(rekClient, collectionId);
rekClient.close();
}

public static void createMyCollection(RekognitionClient rekClient, String
collectionId) {
    try {
        CreateCollectionRequest collectionRequest =
CreateCollectionRequest.builder()
            .collectionId(collectionId)
            .build();

        CreateCollectionResponse collectionResponse =
rekClient.createCollection(collectionRequest);
        System.out.println("CollectionArn: " +
collectionResponse.collectionArn());
        System.out.println("Status code: " +
collectionResponse.statusCode().toString());

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [CreateCollection](#) di Referensi AWS SDK for Java 2.x API.

## Hapus koleksi

Contoh kode berikut menunjukkan cara menghapus koleksi Amazon Rekognition.

Untuk informasi selengkapnya, lihat [Menghapus koleksi](#).

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId>\s

                Where:
                    collectionId - The id of the collection to delete.\s
                """;
```



```

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String collectionId = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Deleting collection: " + collectionId);
    deleteMyCollection(rekClient, collectionId);
    rekClient.close();
}

public static void deleteMyCollection(RekognitionClient rekClient, String
collectionId) {
    try {
        DeleteCollectionRequest deleteCollectionRequest =
DeleteCollectionRequest.builder()
            .collectionId(collectionId)
            .build();

        DeleteCollectionResponse deleteCollectionResponse =
rekClient.deleteCollection(deleteCollectionRequest);
        System.out.println(collectionId + ": " +
deleteCollectionResponse.statusCode().toString());

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [DeleteCollection](#) di Referensi AWS SDK for Java 2.x API.

## Hapus wajah dari koleksi

Contoh kode berikut menunjukkan cara menghapus wajah dari koleksi Rekognition Amazon.

Untuk informasi selengkapnya, lihat [Menghapus wajah dari koleksi](#).

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteFacesFromCollection {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <collectionId> <faceId>\s

            Where:
                collectionId - The id of the collection from which faces are
deleted.\s

                faceId - The id of the face to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
```

```
String faceId = args[1];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

System.out.println("Deleting collection: " + collectionId);
deleteFacesCollection(rekClient, collectionId, faceId);
rekClient.close();
}

public static void deleteFacesCollection(RekognitionClient rekClient,
    String collectionId,
    String faceId) {

    try {
        DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
            .collectionId(collectionId)
            .faceIds(faceId)
            .build();

        rekClient.deleteFaces(deleteFacesRequest);
        System.out.println("The face was deleted from the collection.");

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DeleteFaces](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan koleksi

Contoh kode berikut menunjukkan cara mendeskripsikan koleksi Amazon Rekognition.

Untuk informasi selengkapnya, lihat [Menjelaskan koleksi](#).

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeCollection {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <collectionName>

            Where:
                collectionName - The name of the Amazon Rekognition collection.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();
```

```
        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    public static void describeColl(RekognitionClient rekClient, String
collectionName) {
        try {
            DescribeCollectionRequest describeCollectionRequest =
DescribeCollectionRequest.builder()
                .collectionId(collectionName)
                .build();

            DescribeCollectionResponse describeCollectionResponse = rekClient
                .describeCollection(describeCollectionRequest);
            System.out.println("Collection Arn : " +
describeCollectionResponse.collectionARN());
            System.out.println("Created : " +
describeCollectionResponse.creationTimestamp().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DescribeCollection](#) di Referensi AWS SDK for Java 2.x API.

## Mendeteksi wajah dalam gambar

Contoh kode berikut menunjukkan cara mendeteksi wajah dalam gambar dengan Amazon Rekognition.

Untuk informasi selengkapnya, lihat [Mendeteksi wajah dalam gambar](#).

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.DetectFacesRequest;
import software.amazon.awssdk.services.rekognition.model.DetectFacesResponse;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceDetail;
import software.amazon.awssdk.services.rekognition.model.AgeRange;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectFaces {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
            """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectFacesinImage(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectFacesinImage(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Create an Image object for the source image.
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            DetectFacesRequest facesRequest = DetectFacesRequest.builder()
                .attributes(Attribute.ALL)
                .image(souImage)
                .build();

            DetectFacesResponse facesResponse = rekClient.detectFaces(facesRequest);
            List<FaceDetail> faceDetails = facesResponse.faceDetails();
            for (FaceDetail face : faceDetails) {
                AgeRange ageRange = face.ageRange();
                System.out.println("The detected face is estimated to be between "
                    + ageRange.low().toString() + " and " +
ageRange.high().toString()
                    + " years old.");

                System.out.println("There is a smile : " +
face.smile().value().toString());
            }
        }
    }
}
```

```
        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DetectFaces](#) di Referensi AWS SDK for Java 2.x API.

## Mendeteksi label dalam gambar

Contoh kode berikut menunjukkan cara mendeteksi label dalam gambar dengan Amazon Rekognition.

Untuk informasi selengkapnya, lihat [Mendeteksi label dalam gambar](#).

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsRequest;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```



```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DetectLabels {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectImageLabels(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectImageLabels(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Create an Image object for the source image.
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            DetectLabelsRequest detectLabelsRequest = DetectLabelsRequest.builder()
                .image(souImage)
                .maxLabels(10)
```

```
        .build();

        DetectLabelsResponse labelsResponse =
rekClient.detectLabels(detectLabelsRequest);
        List<Label> labels = labelsResponse.labels();
        System.out.println("Detected labels for the given photo");
        for (Label label : labels) {
            System.out.println(label.name() + ": " +
label.confidence().toString());
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DetectLabels](#) di Referensi AWS SDK for Java 2.x API.

## Mendeteksi label moderasi dalam gambar

Contoh kode berikut menunjukkan cara mendeteksi label moderasi dalam gambar dengan Amazon Rekognition. Label moderasi mengidentifikasi konten yang mungkin tidak pantas untuk beberapa pemirsa.

Untuk informasi selengkapnya, lihat [Mendeteksi gambar yang tidak pantas](#).

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
```

```
import software.amazon.awssdk.services.rekognition.model.Image;
import
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.ModerationLabel;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectModerationLabels {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
            """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectModLabels(rekClient, sourceImage);
        rekClient.close();
    }
}
```

```
}

    public static void detectModLabels(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
                .image(souImage)
                .minConfidence(60F)
                .build();

            DetectModerationLabelsResponse moderationLabelsResponse = rekClient
                .detectModerationLabels(moderationLabelsRequest);
            List<ModerationLabel> labels =
moderationLabelsResponse.moderationLabels();
            System.out.println("Detected labels for image");
            for (ModerationLabel label : labels) {
                System.out.println("Label: " + label.name()
                    + "\n Confidence: " + label.confidence().toString() + "%"
                    + "\n Parent:" + label.parentName());
            }

        } catch (RekognitionException | FileNotFoundException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DetectModerationLabels](#) di Referensi AWS SDK for Java 2.x API.

## Mendeteksi teks dalam gambar

Contoh kode berikut menunjukkan cara mendeteksi teks dalam gambar dengan Amazon Rekognition.

Untuk informasi selengkapnya, lihat [Mendeteksi teks dalam gambar](#).

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DetectTextRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectTextResponse;
import software.amazon.awssdk.services.rekognition.model.TextDetection;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectText {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <sourceImage>

                Where:
                    sourceImage - The path to the image that contains text (for
example, C:\\AWS\\pic1.png).\s
                    """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String sourceImage = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    detectTextLabels(rekClient, sourceImage);
    rekClient.close();
}

public static void detectTextLabels(RekognitionClient rekClient, String
sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectTextRequest textRequest = DetectTextRequest.builder()
            .image(souImage)
            .build();

        DetectTextResponse textResponse = rekClient.detectText(textRequest);
        List<TextDetection> textCollection = textResponse.textDetections();
        System.out.println("Detected lines and words");
        for (TextDetection text : textCollection) {
            System.out.println("Detected: " + text.detectedText());
            System.out.println("Confidence: " + text.confidence().toString());
            System.out.println("Id : " + text.id());
            System.out.println("Parent Id: " + text.parentId());
            System.out.println("Type: " + text.type());
            System.out.println();
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DetectText](#) di Referensi AWS SDK for Java 2.x API.

## Indeks wajah ke koleksi

Contoh kode berikut menunjukkan cara mengindeks wajah dalam gambar dan menambahkannya ke koleksi Rekognition Amazon.

Untuk informasi selengkapnya, lihat [Menambahkan wajah ke koleksi](#).

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.IndexFacesResponse;
import software.amazon.awssdk.services.rekognition.model.IndexFacesRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.QualityFilter;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceRecord;
import software.amazon.awssdk.services.rekognition.model.UnindexedFace;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Reason;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class AddFacesToCollection {
    public static void main(String[] args) {

        final String usage = ""

            Usage:      <collectionId> <sourceImage>

            Where:
                collectionName - The name of the collection.
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        addToCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }

    public static void addToCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            IndexFacesRequest facesRequest = IndexFacesRequest.builder()
                .collectionId(collectionId)
                .image(souImage)
                .maxFaces(1)
```



```

        .qualityFilter(QualityFilter.AUTO)
        .detectionAttributes(Attribute.DEFAULT)
        .build();

    IndexFacesResponse facesResponse = rekClient.indexFaces(facesRequest);
    System.out.println("Results for the image");
    System.out.println("\n Faces indexed:");
    List<FaceRecord> faceRecords = facesResponse.faceRecords();
    for (FaceRecord faceRecord : faceRecords) {
        System.out.println("  Face ID: " + faceRecord.face().faceId());
        System.out.println("  Location:" +
faceRecord.faceDetail().boundingBox().toString());
    }

    List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
    System.out.println("Faces not indexed:");
    for (UnindexedFace unindexedFace : unindexedFaces) {
        System.out.println("  Location:" +
unindexedFace.faceDetail().boundingBox().toString());
        System.out.println("  Reasons:");
        for (Reason reason : unindexedFace.reasons()) {
            System.out.println("Reason: " + reason);
        }
    }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [IndexFaces](#) di Referensi AWS SDK for Java 2.x API.

## Daftar koleksi

Contoh kode berikut menunjukkan cara membuat daftar koleksi Amazon Rekognition.

Untuk informasi selengkapnya, lihat [Daftar koleksi](#).

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsRequest;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCollections {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        System.out.println("Listing collections");
        listAllCollections(rekClient);
        rekClient.close();
    }

    public static void listAllCollections(RekognitionClient rekClient) {
        try {
            ListCollectionsRequest listCollectionsRequest =
ListCollectionsRequest.builder()
                .maxResults(10)
                .build();
```

```

        ListCollectionsResponse response =
rekClient.listCollections(listCollectionsRequest);
        List<String> collectionIds = response.collectionIds();
        for (String resultId : collectionIds) {
            System.out.println(resultId);
        }

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}

```

- Untuk detail API, lihat [ListCollections](#) di Referensi AWS SDK for Java 2.x API.

## Daftar wajah dalam koleksi

Contoh kode berikut menunjukkan cara membuat daftar wajah dalam koleksi Rekognition Amazon.

Untuk informasi selengkapnya, lihat [Daftar wajah dalam koleksi](#).

## SDK for Java 2.x

### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Face;
import software.amazon.awssdk.services.rekognition.model.ListFacesRequest;
import software.amazon.awssdk.services.rekognition.model.ListFacesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.

```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListFacesInCollection {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <collectionId>

            Where:
                collectionId - The name of the collection.\s
            "";

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        System.out.println("Faces in collection " + collectionId);
        listFacesCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void listFacesCollection(RekognitionClient rekClient, String
collectionId) {
        try {
            ListFacesRequest facesRequest = ListFacesRequest.builder()
                .collectionId(collectionId)
                .maxResults(10)
                .build();

            ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
            List<Face> faces = facesResponse.faces();
            for (Face face : faces) {
                System.out.println("Confidence level there is a face: " +
face.confidence());
            }
        }
    }
}
```

```
        System.out.println("The face Id value is " + face.faceId());
    }

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [ListFaces](#) di Referensi AWS SDK for Java 2.x API.

Kenali selebriti dalam sebuah gambar

Contoh kode berikut menunjukkan cara mengenali selebriti dalam gambar dengan Amazon Rekognition.

Untuk informasi selengkapnya, lihat [Mengenali selebriti dalam sebuah gambar](#).

SDK for Java 2.x

**Note**

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesRequest;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Celebrity;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RecognizeCelebrities {
    public static void main(String[] args) {
        final String usage = ""
            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png)\\.\\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        System.out.println("Locating celebrities in " + sourceImage);
        recognizeAllCelebrities(rekClient, sourceImage);
        rekClient.close();
    }

    public static void recognizeAllCelebrities(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();
        }
    }
}
```

```
        RecognizeCelebritiesRequest request =
RecognizeCelebritiesRequest.builder()
        .image(souImage)
        .build();

        RecognizeCelebritiesResponse result =
rekClient.recognizeCelebrities(request);
        List<Celebrity> celebs = result.celebrityFaces();
        System.out.println(celebs.size() + " celebrity(s) were recognized.\n");
        for (Celebrity celebrity : celebs) {
            System.out.println("Celebrity recognized: " + celebrity.name());
            System.out.println("Celebrity ID: " + celebrity.id());

            System.out.println("Further information (if available):");
            for (String url : celebrity.urls()) {
                System.out.println(url);
            }
            System.out.println();
        }
        System.out.println(result.unrecognizedFaces().size() + " face(s) were
unrecognized.");

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [RecognizeCelebrities](#) di Referensi AWS SDK for Java 2.x API.

## Cari wajah dalam koleksi

Contoh kode berikut menunjukkan cara mencari wajah dalam koleksi Rekognition Amazon yang cocok dengan wajah lain dari koleksi.

Untuk informasi selengkapnya, lihat [Mencari wajah \(ID wajah\)](#).

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingImageCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <sourceImage>

                Where:
                    collectionId - The id of the collection. \s
                    sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\s

                """;
```



```
    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String collectionId = args[0];
    String sourceImage = args[1];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Searching for a face in a collections");
    searchFaceInCollection(rekClient, collectionId, sourceImage);
    rekClient.close();
}

public static void searchFaceInCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(new File(sourceImage));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        SearchFacesByImageRequest facesByImageRequest =
SearchFacesByImageRequest.builder()
            .image(souImage)
            .maxFaces(10)
            .faceMatchThreshold(70F)
            .collectionId(collectionId)
            .build();

        SearchFacesByImageResponse imageResponse =
rekClient.searchFacesByImage(facesByImageRequest);
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face : faceImageMatches) {
            System.out.println("The similarity level is " + face.similarity());
            System.out.println();
        }
    }
}
```

```
        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [SearchFaces](#) di Referensi AWS SDK for Java 2.x API.

Cari wajah dalam koleksi dibandingkan dengan gambar referensi

Contoh kode berikut menunjukkan cara mencari wajah dalam koleksi Rekognition Amazon dibandingkan dengan gambar referensi.

Untuk informasi selengkapnya, lihat [Mencari wajah \(gambar\)](#).

SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingIdCollection {
```

```
public static void main(String[] args) {
    final String usage = ""

        Usage:    <collectionId> <sourceImage>

        Where:
            collectionId - The id of the collection. \s
            sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String collectionId = args[0];
    String faceId = args[1];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Searching for a face in a collections");
    searchFaceById(rekClient, collectionId, faceId);
    rekClient.close();
}

public static void searchFaceById(RekognitionClient rekClient, String
collectionId, String faceId) {
    try {
        SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
            .collectionId(collectionId)
            .faceId(faceId)
            .faceMatchThreshold(70F)
            .maxFaces(2)
            .build();

        SearchFacesResponse imageResponse =
rekClient.searchFaces(searchFacesRequest);
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face : faceImageMatches) {
            System.out.println("The similarity level is " + face.similarity());
        }
    }
}
```

```
        System.out.println();
    }

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [SearchFacesByImage](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

### Mendeteksi informasi dalam video

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Mulai pekerjaan Amazon Rekognition untuk mendeteksi elemen seperti orang, objek, dan teks dalam video.
- Periksa status pekerjaan sampai pekerjaan selesai.
- Output daftar elemen yang terdeteksi oleh setiap pekerjaan.

### SDK for Java 2.x

#### Note

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

Dapatkan hasil selebriti dari video yang terletak di ember Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
```

```

import
    software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
import
    software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;

/**
 * To run this code example, ensure that you perform the Prerequisites as stated
 * in the Amazon Rekognition Guide:
 * https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-sqs.html
 *
 * Also, ensure that set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class VideoCelebrityDetection {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
                (for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
                (Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
                role to use.\s

```

```
        """);

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startCelebrityDetection(rekClient, channel, bucket, video);
    getCelebrityDetectionResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

public static void startCelebrityDetection(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
            .jobTag("Celebrities")
```

```
        .notificationChannel(channel)
        .video(vidOb)
        .build();

    StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient
        .startCelebrityRecognition(recognitionRequest);
    startJobId = startCelebrityRecognitionResult.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void getCelebrityDetectionResults(RekognitionClient rekClient) {

    try {
        String paginationToken = null;
        GetCelebrityRecognitionResponse recognitionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (recognitionResponse != null)
                paginationToken = recognitionResponse.nextToken();

            GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                .maxResults(10)
                .build();

            // Wait until the job succeeds
            while (!finished) {
                recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
                status = recognitionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
            }
        }
    }
}
```

```

        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null.
    VideoMetadata videoMetaData = recognitionResponse.videoMetadata();
    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " + videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());
    System.out.println("Job");

    List<CelebrityRecognition> celebs =
recognitionResponse.celebrities();
    for (CelebrityRecognition celeb : celebs) {
        long seconds = celeb.timestamp() / 1000;
        System.out.print("Sec: " + seconds + " ");
        CelebrityDetail details = celeb.celebrity();
        System.out.println("Name: " + details.name());
        System.out.println("Id: " + details.id());
        System.out.println();
    }

    } while (recognitionResponse.nextToken() != null);

} catch (RecognitionException | InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
}
}

```

Mendeteksi label dalam video dengan operasi deteksi label.

```

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;

```



```
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
                (for example, (for example, myBucket).\s
                video - The name of the video (for example, people.mp4).\s
```

```
        queueUrl- The URL of a SQS queue.\s
        topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
        roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
        """;

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String queueUrl = args[2];
    String topicArn = args[3];
    String roleArn = args[4];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    SqsClient sqs = SqsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startLabels(rekClient, channel, bucket, video);
    getLabelJob(rekClient, sqs, queueUrl);
    System.out.println("This example is done!");
    sqs.close();
    rekClient.close();
}

public static void startLabels(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3Obj = S3Object.builder()
```

```
        .bucket(bucket)
        .name(video)
        .build();

    Video vid0b = Video.builder()
        .s3Object(s3obj)
        .build();

    StartLabelDetectionRequest labelDetectionRequest =
    StartLabelDetectionRequest.builder()
        .jobTag("DetectingLabels")
        .notificationChannel(channel)
        .video(vid0b)
        .minConfidence(50F)
        .build();

    StartLabelDetectionResponse labelDetectionResponse =
    rekClient.startLabelDetection(labelDetectionRequest);
    startJobId = labelDetectionResponse.jobId();

    boolean ans = true;
    String status = "";
    int yy = 0;
    while (ans) {

        GetLabelDetectionRequest detectionRequest =
    GetLabelDetectionRequest.builder()
        .jobId(startJobId)
        .maxResults(10)
        .build();

        GetLabelDetectionResponse result =
    rekClient.getLabelDetection(detectionRequest);
        status = result.jobStatusAsString();

        if (status.compareTo("SUCCEEDED") == 0)
            ans = false;
        else
            System.out.println(yy + " status is: " + status);

        Thread.sleep(1000);
        yy++;
    }
}
```

```
        System.out.println(startJobId + " status is: " + status);

    } catch (RekognitionException | InterruptedException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();

        if (!messages.isEmpty()) {
            for (Message message : messages) {
                String notification = message.body();

                // Get the status and job id from the notification
                ObjectMapper mapper = new ObjectMapper();
                JsonNode jsonMessageTree = mapper.readTree(notification);
                JsonNode messageBodyText = jsonMessageTree.get("Message");
                ObjectMapper operationResultMapper = new ObjectMapper();
                JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                JsonNode operationJobId = jsonResultTree.get("JobId");
                JsonNode operationStatus = jsonResultTree.get("Status");
                System.out.println("Job found in JSON is " + operationJobId);

                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

                String jobId = operationJobId.textValue();
                if (startJobId.compareTo(jobId) == 0) {
                    System.out.println("Job id: " + operationJobId);
                    System.out.println("Status : " +
operationStatus.toString());
                }
            }
        }
    }
}
```

```

        if (operationStatus.asText().equals("SUCCEEDED"))
            getResultsLabels(rekClient);
        else
            System.out.println("Video analysis failed");

        sqs.deleteMessage(deleteMessageRequest);
    } else {
        System.out.println("Job received was not job " +
startJobId);
        sqs.deleteMessage(deleteMessageRequest);
    }
}

} catch (RekognitionException e) {
    e.getMessage();
    System.exit(1);
} catch (JsonMappingException e) {
    e.printStackTrace();
} catch (JsonProcessingException e) {
    e.printStackTrace();
}
}

// Gets the job results by calling GetLabelDetection
private static void getResultsLabels(RekognitionClient rekClient) {

    int maxResults = 10;
    String paginationToken = null;
    GetLabelDetectionResponse labelDetectionResult = null;

    try {
        do {
            if (labelDetectionResult != null)
                paginationToken = labelDetectionResult.nextToken();

            GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .sortBy(LabelDetectionSortBy.TIMESTAMP)
                .maxResults(maxResults)
                .nextToken(paginationToken)
                .build();

```

```
        labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
        VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " + videoMetaData.durationMillis());
        System.out.println("FrameRate: " + videoMetaData.frameRate());

        List<LabelDetection> detectedLabels = labelDetectionResult.labels();
        for (LabelDetection detectedLabel : detectedLabels) {
            long seconds = detectedLabel.timestamp();
            Label label = detectedLabel.label();
            System.out.println("Millisecond: " + seconds + " ");

            System.out.println("    Label:" + label.name());
            System.out.println("    Confidence:" +
detectedLabel.label().confidence().toString());

            List<Instance> instances = label.instances();
            System.out.println("    Instances of " + label.name());

            if (instances.isEmpty()) {
                System.out.println("        " + "None");
            } else {
                for (Instance instance : instances) {
                    System.out.println("        Confidence: " +
instance.confidence().toString());
                    System.out.println("        Bounding box: " +
instance.boundingBox().toString());
                }
            }
            System.out.println("    Parent labels for " + label.name() +
":");

            List<Parent> parents = label.parents();

            if (parents.isEmpty()) {
                System.out.println("        None");
            } else {
                for (Parent parent : parents) {
                    System.out.println("        " + parent.name());
                }
            }
            System.out.println();
        }
    }
```

```

        } while (labelDetectionResult != null &&
labelDetectionResult.nextToken() != null);

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
}

```

## Mendeteksi wajah dalam video yang disimpan dalam bucket Amazon S3.

```

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.

```

```

*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of the video (for example, people.mp4).\s
                queueUrl- The URL of a SQS queue.\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        SqsClient sqs = SqsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        NotificationChannel channel = NotificationChannel.builder()

```



```
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

startLabels(rekClient, channel, bucket, video);
getLabelJob(rekClient, sqs, queueUrl);
System.out.println("This example is done!");
sqs.close();
rekClient.close();
}

public static void startLabels(RecognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidObj)
            .minConfidence(50F)
            .build();

        StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

        boolean ans = true;
        String status = "";
        int yy = 0;
        while (ans) {

            GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
```

```
        .jobId(startJobId)
        .maxResults(10)
        .build();

    GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
    status = result.jobStatusAsString();

    if (status.compareTo("SUCCEEDED") == 0)
        ans = false;
    else
        System.out.println(yy + " status is: " + status);

    Thread.sleep(1000);
    yy++;
}

System.out.println(startJobId + " status is: " + status);

} catch (RekognitionException | InterruptedException e) {
    e.getMessage();
    System.exit(1);
}
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();

        if (!messages.isEmpty()) {
            for (Message message : messages) {
                String notification = message.body();

                // Get the status and job id from the notification
                ObjectMapper mapper = new ObjectMapper();
                JsonNode jsonMessageTree = mapper.readTree(notification);
                JsonNode messageBodyText = jsonMessageTree.get("Message");
                ObjectMapper operationResultMapper = new ObjectMapper();
```

```
        JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
        JsonNode operationJobId = jsonResultTree.get("JobId");
        JsonNode operationStatus = jsonResultTree.get("Status");
        System.out.println("Job found in JSON is " + operationJobId);

        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

        String jobId = operationJobId.textValue();
        if (startJobId.compareTo(jobId) == 0) {
            System.out.println("Job id: " + operationJobId);
            System.out.println("Status : " +
operationStatus.toString());

            if (operationStatus.asText().equals("SUCCEEDED"))
                getResultsLabels(rekClient);
            else
                System.out.println("Video analysis failed");

            sqs.deleteMessage(deleteMessageRequest);
        } else {
            System.out.println("Job received was not job " +
startJobId);

            sqs.deleteMessage(deleteMessageRequest);
        }
    }
}

} catch (RekognitionException e) {
    e.getMessage();
    System.exit(1);
} catch (JsonMappingException e) {
    e.printStackTrace();
} catch (JsonProcessingException e) {
    e.printStackTrace();
}
}

// Gets the job results by calling GetLabelDetection
private static void getResultsLabels(RekognitionClient rekClient) {
```

```
int maxResults = 10;
String paginationToken = null;
GetLabelDetectionResponse labelDetectionResult = null;

try {
    do {
        if (labelDetectionResult != null)
            paginationToken = labelDetectionResult.nextToken();

        GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
            .jobId(startJobId)
            .sortBy(LabelDetectionSortBy.TIMESTAMP)
            .maxResults(maxResults)
            .nextToken(paginationToken)
            .build();

        labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
        VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " + videoMetaData.durationMillis());
        System.out.println("FrameRate: " + videoMetaData.frameRate());

        List<LabelDetection> detectedLabels = labelDetectionResult.labels();
        for (LabelDetection detectedLabel : detectedLabels) {
            long seconds = detectedLabel.timestamp();
            Label label = detectedLabel.label();
            System.out.println("Millisecond: " + seconds + " ");

            System.out.println("  Label:" + label.name());
            System.out.println("  Confidence:" +
detectedLabel.label().confidence().toString());

            List<Instance> instances = label.instances();
            System.out.println("  Instances of " + label.name());

            if (instances.isEmpty()) {
                System.out.println("    " + "None");
            } else {
                for (Instance instance : instances) {
                    System.out.println("    Confidence: " +
instance.confidence().toString());
                }
            }
        }
    }
}
```

```

                System.out.println("        Bounding box: " +
instance.boundingBox().toString());
            }
        }
        System.out.println("    Parent labels for " + label.name() +
");");
        List<Parent> parents = label.parents();

        if (parents.isEmpty()) {
            System.out.println("        None");
        } else {
            for (Parent parent : parents) {
                System.out.println("        " + parent.name());
            }
        }
        System.out.println();
    }
} while (labelDetectionResult != null &&
labelDetectionResult.nextToken() != null);

    } catch (RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}
}

```

Mendeteksi konten yang tidak pantas atau menyinggung dalam video yang disimpan di bucket Amazon S3.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;

```

```
import
    software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectInappropriate {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
```

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

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startModerationDetection(rekClient, channel, bucket, video);
    getModResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

public static void startModerationDetection(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {

    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3obj)
            .build();

        StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
            .jobTag("Moderation")
            .notificationChannel(channel)
            .video(vidObj)
            .build();

        StartContentModerationResponse startModDetectionResult = rekClient
            .startContentModeration(modDetectionRequest);
        startJobId = startModDetectionResult.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}

public static void getModResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetContentModerationResponse modDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (modDetectionResponse != null)
                paginationToken = modDetectionResponse.nextToken();

            GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {
                modDetectionResponse =
rekClient.getContentModeration(modRequest);
                status = modDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }

            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null.
            VideoMetadata videoMetaData = modDetectionResponse.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());
        }
    }
}
```



```

        System.out.println("Job");

        List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
        for (ContentModerationDetection mod : mods) {
            long seconds = mod.timestamp() / 1000;
            System.out.print("Mod label: " + seconds + " ");
            System.out.println(mod.moderationLabel().toString());
            System.out.println();
        }

        } while (modDetectionResponse != null &&
modDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}

```

Mendeteksi segmen isyarat teknis dan segmen deteksi bidikan dalam video yang disimpan dalam bucket Amazon S3.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartShotDetectionFilter;
import
    software.amazon.awssdk.services.rekognition.model.StartTechnicalCueDetectionFilter;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionFilters;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionRequest;

```

```
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.SegmentDetection;
import software.amazon.awssdk.services.rekognition.model.TechnicalCueSegment;
import software.amazon.awssdk.services.rekognition.model.ShotSegment;
import software.amazon.awssdk.services.rekognition.model.SegmentType;
import software.amazon.awssdk.services.sqs.SqsClient;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectSegment {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
    }
}
```

```
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

SqsClient sqs = SqsClient.builder()
    .region(Region.US_EAST_1)
    .build();

NotificationChannel channel = NotificationChannel.builder()
    .snsTopicArn(topicArn)
    .roleArn(roleArn)
    .build();

startSegmentDetection(rekClient, channel, bucket, video);
getSegmentResults(rekClient);
System.out.println("This example is done!");
sqs.close();
rekClient.close();
}

public static void startSegmentDetection(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3obj)
            .build();

        StartShotDetectionFilter cueDetectionFilter =
StartShotDetectionFilter.builder()
            .minSegmentConfidence(60F)
            .build();

        StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
StartTechnicalCueDetectionFilter.builder()
            .minSegmentConfidence(60F)
            .build();
```

```
        StartSegmentDetectionFilters filters =
StartSegmentDetectionFilters.builder()
        .shotFilter(cueDetectionFilter)
        .technicalCueFilter(technicalCueDetectionFilter)
        .build();

        StartSegmentDetectionRequest segDetectionRequest =
StartSegmentDetectionRequest.builder()
        .jobTag("DetectingLabels")
        .notificationChannel(channel)
        .segmentTypes(SegmentType.TECHNICAL_CUE, SegmentType.SHOT)
        .video(vidOb)
        .filters(filters)
        .build();

        StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
        startJobId = segDetectionResponse.jobId();

    } catch (RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getSegmentResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetSegmentDetectionResponse segDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (segDetectionResponse != null)
                paginationToken = segDetectionResponse.nextToken();

            GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();
```

```
        // Wait until the job succeeds.
        while (!finished) {
            segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
            status = segDetectionResponse.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(yy + " status is: " + status);
                Thread.sleep(1000);
            }
            yy++;
        }
        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is null.
        List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
        for (VideoMetadata metaData : videoMetaData) {
            System.out.println("Format: " + metaData.format());
            System.out.println("Codec: " + metaData.codec());
            System.out.println("Duration: " + metaData.durationMillis());
            System.out.println("FrameRate: " + metaData.frameRate());
            System.out.println("Job");
        }

        List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
        for (SegmentDetection detectedSegment : detectedSegments) {
            String type = detectedSegment.type().toString();
            if (type.contains(SegmentType.technical_cue.toString())) {
                System.out.println("Technical Cue");
                TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
                System.out.println("\tType: " + segmentCue.type());
                System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
            }

            if (type.contains(SegmentType.shot.toString())) {
                System.out.println("Shot");
                ShotSegment segmentShot = detectedSegment.shotSegment();
```

```

                System.out.println("\tIndex " + segmentShot.index());
                System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
            }

            long seconds = detectedSegment.durationMillis();
            System.out.println("\tDuration : " + seconds + " milliseconds");
            System.out.println("\tStart time code: " +
detectedSegment.startTimecodeSMPTE());
            System.out.println("\tEnd time code: " +
detectedSegment.endTimecodeSMPTE());
            System.out.println("\tDuration time code: " +
detectedSegment.durationSMPTE());
            System.out.println();
        }

        } while (segDetectionResponse != null &&
segDetectionResponse.nextToken() != null);

    } catch (RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}

```

Mendeteksi teks dalam video yang disimpan dalam video yang disimpan dalam bucket Amazon S3.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;

```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectText {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        NotificationChannel channel = NotificationChannel.builder()
            .snsTopicArn(topicArn)
```

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

    startTextLabels(rekClient, channel, bucket, video);
    getTextResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

public static void startTextLabels(RecognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vid0b = Video.builder()
            .s3Object(s3obj)
            .build();

        StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vid0b)
            .build();

        StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

    } catch (RecognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void getTextResults(RecognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetTextDetectionResponse textDetectionResponse = null;
```



```
boolean finished = false;
String status;
int yy = 0;

do {
    if (textDetectionResponse != null)
        paginationToken = textDetectionResponse.nextToken();

    GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
        .jobId(startJobId)
        .nextToken(paginationToken)
        .maxResults(10)
        .build();

    // Wait until the job succeeds.
    while (!finished) {
        textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
        status = textDetectionResponse.jobStatusAsString();

        if (status.compareTo("SUCCEEDED") == 0)
            finished = true;
        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null.
    VideoMetadata videoMetaData = textDetectionResponse.videoMetadata();
    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " + videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());
    System.out.println("Job");

    List<TextDetectionResult> labels =
textDetectionResponse.textDetections();
    for (TextDetectionResult detectedText : labels) {
```

```

        System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
        System.out.println("Id : " + detectedText.textDetection().id());
        System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
        System.out.println("Type: " +
detectedText.textDetection().type());
        System.out.println("Text: " +
detectedText.textDetection().detectedText());
        System.out.println();
    }

    } while (textDetectionResponse != null &&
textDetectionResponse.nextToken() != null);

    } catch (RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

Mendeteksi orang dalam video yang disimpan dalam video yang disimpan dalam bucket Amazon S3.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development

```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class VideoPersonDetection {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        NotificationChannel channel = NotificationChannel.builder()
            .snsTopicArn(topicArn)
            .roleArn(roleArn)
            .build();
```

```
        startPersonLabels(rekClient, channel, bucket, video);
        getPersonDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startPersonLabels(RecognitionClient rekClient,
        NotificationChannel channel,
        String bucket,
        String video) {
        try {
            S3Object s3obj = S3Object.builder()
                .bucket(bucket)
                .name(video)
                .build();

            Video vid0b = Video.builder()
                .s3object(s3obj)
                .build();

            StartPersonTrackingRequest personTrackingRequest =
                StartPersonTrackingRequest.builder()
                    .jobTag("DetectingLabels")
                    .video(vid0b)
                    .notificationChannel(channel)
                    .build();

            StartPersonTrackingResponse labelDetectionResponse =
                rekClient.startPersonTracking(personTrackingRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RecognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getPersonDetectionResults(RecognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetPersonTrackingResponse personTrackingResult = null;
            boolean finished = false;
            String status;
            int yy = 0;
        }
    }
}
```

```
do {
    if (personTrackingResult != null)
        paginationToken = personTrackingResult.nextToken();

    GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
        .jobId(startJobId)
        .nextToken(paginationToken)
        .maxResults(10)
        .build();

    // Wait until the job succeeds
    while (!finished) {

        personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
        status = personTrackingResult.jobStatusAsString();

        if (status.compareTo("SUCCEEDED") == 0)
            finished = true;
        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null.
    VideoMetadata videoMetaData = personTrackingResult.videoMetadata();

    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " + videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());
    System.out.println("Job");

    List<PersonDetection> detectedPersons =
personTrackingResult.persons();
    for (PersonDetection detectedPerson : detectedPersons) {
        long seconds = detectedPerson.timestamp() / 1000;
        System.out.print("Sec: " + seconds + " ");
    }
}
```

```
                System.out.println("Person Identifier: " +
detectedPerson.person().index());
                System.out.println();
            }

            } while (personTrackingResult != null &&
personTrackingResult.nextToken() != null);

            } catch (RekognitionException | InterruptedException e) {
                System.out.println(e.getMessage());
                System.exit(1);
            }
        }
    }
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [GetCelebrityRecognition](#)
  - [GetContentModeration](#)
  - [GetLabelDetection](#)
  - [GetPersonTracking](#)
  - [GetSegmentDetection](#)
  - [GetTextDetection](#)
  - [StartCelebrityRecognition](#)
  - [StartContentModeration](#)
  - [StartLabelDetection](#)
  - [StartPersonTracking](#)
  - [StartSegmentDetection](#)
  - [StartTextDetection](#)

## Route 53 contoh pendaftaran domain menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan pendaftaran domain AWS SDK for Java 2.x with Route 53.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

Setiap contoh menyertakan tautan ke GitHub, di mana Anda dapat menemukan petunjuk 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 for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.route53domains.Route53DomainsClient;
import software.amazon.awssdk.services.route53.model.Route53Exception;
import software.amazon.awssdk.services.route53domains.model.DomainPrice;
import software.amazon.awssdk.services.route53domains.model.ListPricesRequest;
import software.amazon.awssdk.services.route53domains.model.ListPricesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code examples performs the following operation:
 *
 */
```

```
* 1. Invokes ListPrices for at least one domain type, such as the "com" type
* and displays the prices for Registration and Renewal.
*
*/
public class HelloRoute53 {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <hostedZoneId> \n\n" +
            "Where:\n" +
            "    hostedZoneId - The id value of an existing hosted zone. \n";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainType = args[0];
        Region region = Region.US_EAST_1;
        Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("Invokes ListPrices for at least one domain type.");
        listPrices(route53DomainsClient, domainType);
        System.out.println(DASHES);
    }

    public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                .maxItems(10)
                .tld(domainType)
                .build();

            ListPricesResponse response =
route53DomainsClient.listPrices(pricesRequest);
            List<DomainPrice> prices = response.prices();
            for (DomainPrice pr : prices) {
                System.out.println("Name: " + pr.name());
            }
        }
    }
}
```



```
        System.out.println(
            "Registration: " + pr.registrationPrice().price() + " " +
pr.registrationPrice().currency());
        System.out.println("Renewal: " + pr.renewalPrice().price() + " " +
pr.renewalPrice().currency());
        System.out.println("Transfer: " + pr.transferPrice().price() + " " +
pr.transferPrice().currency());
        System.out.println("Transfer: " + pr.transferPrice().price() + " " +
pr.transferPrice().currency());
        System.out.println("Change Ownership: " +
pr.changeOwnershipPrice().price() + " "
            + pr.changeOwnershipPrice().currency());
        System.out.println(
            "Restoration: " + pr.restorationPrice().price() + " " +
pr.restorationPrice().currency());
        System.out.println(" ");
    }

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [ListPrices](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

Periksa ketersediaan domain

Contoh kode berikut menunjukkan cara memeriksa ketersediaan domain.

## SDK for Java 2.x

 Note

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

```
public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainAvailabilityResponse response = route53DomainsClient
            .checkDomainAvailability(availabilityRequest);
        System.out.println(domainSuggestion + " is " +
response.availability().toString());


    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [CheckDomainAvailability](#) di Referensi AWS SDK for Java 2.x API.

## Periksa transferabilitas domain

Contoh kode berikut menunjukkan cara memeriksa transferabilitas domain.

## SDK for Java 2.x

 Note

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

```
public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainTransferabilityResponse response = route53DomainsClient
            .checkDomainTransferability(transferabilityRequest);
        System.out.println("Transferability: " +
response.transferability().transferable().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [CheckDomainTransferability](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan detail domain

Contoh kode berikut menunjukkan cara mendapatkan detail untuk domain.

SDK for Java 2.x

#### Note

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

```
public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
    try {
        GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
            .domainName(domainSuggestion)
            .build();
```

```
        GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
        System.out.println("The contact first name is " +
response.registrantContact().firstName());
        System.out.println("The contact last name is " +
response.registrantContact().lastName());
        System.out.println("The contact org name is " +
response.registrantContact().organizationName());


    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [GetDomainDetail](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan detail operasi

Contoh kode berikut menunjukkan bagaimana untuk mendapatkan rincian tentang operasi.

SDK for Java 2.x

 Note

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

```
public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
    try {
        GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
            .operationId(operationId)
            .build();

        GetOperationDetailResponse response =
route53DomainsClient.getOperationDetail(detailRequest);
        System.out.println("Operation detail message is " + response.message());
    }
}
```


```
    } catch (Route53Exception e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- Untuk detail API, lihat [GetOperationDetail](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan nama domain yang disarankan

Contoh kode berikut menunjukkan cara mendapatkan saran nama domain.

SDK for Java 2.x

 Note

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

```
public static void listDomainSuggestions(Route53DomainsClient  
route53DomainsClient, String domainSuggestion) {  
    try {  
        GetDomainSuggestionsRequest suggestionsRequest =  
GetDomainSuggestionsRequest.builder()  
            .domainName(domainSuggestion)  
            .suggestionCount(5)  
            .onlyAvailable(true)  
            .build();  
  
        GetDomainSuggestionsResponse response =  
route53DomainsClient.getDomainSuggestions(suggestionsRequest);  
        List<DomainSuggestion> suggestions = response.suggestionsList();  
        for (DomainSuggestion suggestion : suggestions) {  
            System.out.println("Suggestion Name: " + suggestion.domainName());  
            System.out.println("Availability: " + suggestion.availability());  
            System.out.println(" ");  
        }  
    }  
}
```

```
    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [GetDomainSuggestions](#) di Referensi AWS SDK for Java 2.x API.

## Daftar harga domain

Contoh kode berikut menunjukkan bagaimana untuk daftar harga domain.

### SDK for Java 2.x

#### Note

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

```
public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
    try {
        ListPricesRequest pricesRequest = ListPricesRequest.builder()
            .tld(domainType)
            .build();

        ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
        listRes.stream()
            .flatMap(r -> r.prices().stream())
            .forEach(content -> System.out.println(" Name: " +
content.name() +
                " Registration: " + content.registrationPrice().price()
+ " "
                + content.registrationPrice().currency() +
                " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ListPrices](#) di Referensi AWS SDK for Java 2.x API.

## Daftar domain

Contoh kode berikut menunjukkan bagaimana untuk daftar domain terdaftar.

### SDK for Java 2.x

#### Note

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

```
public static void listDomains(Route53DomainsClient route53DomainsClient) {
    try {
        ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
        listRes.stream()
            .flatMap(r -> r.domains().stream())
            .forEach(content -> System.out.println("The domain name is " +
content.domainName()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ListDomains](#) di Referensi AWS SDK for Java 2.x API.

## Daftar operasi

Contoh kode berikut menunjukkan bagaimana untuk daftar operasi.

## SDK for Java 2.x

 Note

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

```
public static void listOperations(Route53DomainsClient route53DomainsClient) {
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        localDateTime = localDateTime.minusYears(1);
        Instant myTime = localDateTime.toInstant(zoneOffset);

        ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
            .submittedSince(myTime)
            .build();

        ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
        listRes.stream()
            .flatMap(r -> r.operations().stream())
            .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                " Status: " + content.statusAsString() +
                " Date: " + content.submittedDate()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ListOperations](#) di Referensi AWS SDK for Java 2.x API.



## Mendaftarkan domain

Contoh kode berikut menunjukkan cara mendaftarkan domain.

### SDK for Java 2.x

#### Note

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

```
public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
    String domainSuggestion,
    String phoneNumber,
    String email,
    String firstName,
    String lastName,
    String city) {

    try {
        ContactDetail contactDetail = ContactDetail.builder()
            .contactType(ContactType.COMPANY)
            .state("LA")
            .countryCode(CountryCode.IN)
            .email(email)
            .firstName(firstName)
            .lastName(lastName)
            .city(city)
            .phoneNumber(phoneNumber)
            .organizationName("My Org")
            .addressLine1("My Address")
            .zipCode("123 123")
            .build();

        RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
            .adminContact(contactDetail)
            .registrantContact(contactDetail)
            .techContact(contactDetail)
            .domainName(domainSuggestion)
            .autoRenew(true)
            .durationInYears(1)
```

```
        .build();

        RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
        System.out.println("Registration requested. Operation Id: " +
response.operationId());
        return response.operationId();

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Untuk detail API, lihat [RegisterDomain](#) di Referensi AWS SDK for Java 2.x API.

## Tampilkan penagihan

Contoh kode berikut menunjukkan cara melihat catatan penagihan.

## SDK for Java 2.x

### Note

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

```
public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        LocalDateTime localDateTime2 = localDateTime.minusYears(1);
        Instant myStartTime = localDateTime2.toInstant(zoneOffset);
        Instant myEndTime = localDateTime.toInstant(zoneOffset);
```

```
ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
    .start(myStartTime)
    .end(myEndTime)
    .build();

ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
listRes.stream()
    .flatMap(r -> r.billingRecords().stream())
    .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
        " Operation: " + content.operationAsString() +
        " Price: " + content.price()));

} catch (Route53Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Untuk detail API, lihat [ViewBilling](#) di Referensi AWS SDK for Java 2.x API.


## Skenario

### Memulai dengan domain

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- 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 for Java 2.x

 Note

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

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example uses pagination methods where applicable. For example, to list
 * domains, the
 * listDomainsPaginator method is used. For more information about pagination,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/pagination.html
 *
 * This Java code example performs the following operations:
 *
 * 1. List current domains.
 * 2. List operations in the past year.
 * 3. View billing for the account in the past year.
 * 4. View prices for domain types.
 * 5. Get domain suggestions.
 * 6. Check domain availability.
 * 7. Check domain transferability.
 * 8. Request a domain registration.
 * 9. Get operation details.
 * 10. Optionally, get domain details.
 */

public class Route53Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = ""
```

```

Usage:
    <domainType> <phoneNumber> <email> <domainSuggestion>
<firstName> <lastName> <city>

Where:
    domainType - The domain type (for example, com).\s
    phoneNumber - The phone number to use (for example,
+91.9966564xxx)    email - The email address to use.    domainSuggestion - The
domain suggestion (for example, findmy.accountants).\s
    firstName - The first name to use to register a domain.\s
    lastName - The last name to use to register a domain.\s
    city - the city to use to register a domain.\s
    """;

if (args.length != 7) {
    System.out.println(usage);
    System.exit(1);
}

String domainType = args[0];
String phoneNumber = args[1];
String email = args[2];
String domainSuggestion = args[3];
String firstName = args[4];
String lastName = args[5];
String city = args[6];
Region region = Region.US_EAST_1;
Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon Route 53 domains example
scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. List current domains.");
listDomains(route53DomainsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. List operations in the past year.");

```

```
listOperations(route53DomainsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. View billing for the account in the past year.");
listBillingRecords(route53DomainsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. View prices for domain types.");
listPrices(route53DomainsClient, domainType);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get domain suggestions.");
listDomainSuggestions(route53DomainsClient, domainSuggestion);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Check domain availability.");
checkDomainAvailability(route53DomainsClient, domainSuggestion);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Check domain transferability.");
checkDomainTransferability(route53DomainsClient, domainSuggestion);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Request a domain registration.");
String opId = requestDomainRegistration(route53DomainsClient,
    domainSuggestion, phoneNumber, email, firstName,
    lastName, city);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Get operation details.");
getOperationalDetail(route53DomainsClient, opId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get domain details.");
System.out.println("Note: You must have a registered domain to get
details.");
```

```
        System.out.println("Otherwise, an exception is thrown that states ");
        System.out.println("Domain xxxxxxxx not found in xxxxxxxx account.");
        getDomainDetails(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);
    }

    public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
        try {
            GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
                .domainName(domainSuggestion)
                .build();

            GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
            System.out.println("The contact first name is " +
response.registrantContact().firstName());
            System.out.println("The contact last name is " +
response.registrantContact().lastName());
            System.out.println("The contact org name is " +
response.registrantContact().organizationName());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
        try {
            GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
                .operationId(operationId)
                .build();

            GetOperationDetailResponse response =
route53DomainsClient.getOperationDetail(detailRequest);
            System.out.println("Operation detail message is " + response.message());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

```
}

    public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
        String domainSuggestion,
        String phoneNumber,
        String email,
        String firstName,
        String lastName,
        String city) {

    try {
        ContactDetail contactDetail = ContactDetail.builder()
            .contactType(ContactType.COMPANY)
            .state("LA")
            .countryCode(CountryCode.IN)
            .email(email)
            .firstName(firstName)
            .lastName(lastName)
            .city(city)
            .phoneNumber(phoneNumber)
            .organizationName("My Org")
            .addressLine1("My Address")
            .zipCode("123 123")
            .build();

        RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
            .adminContact(contactDetail)
            .registrantContact(contactDetail)
            .techContact(contactDetail)
            .domainName(domainSuggestion)
            .autoRenew(true)
            .durationInYears(1)
            .build();

        RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
        System.out.println("Registration requested. Operation Id: " +
response.operationId());
        return response.operationId();

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```



```
    }
    return "";
}

public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainTransferabilityResponse response = route53DomainsClient
            .checkDomainTransferability(transferabilityRequest);
        System.out.println("Transferability: " +
response.transferability().transferable().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainAvailabilityResponse response = route53DomainsClient
            .checkDomainAvailability(availabilityRequest);
        System.out.println(domainSuggestion + " is " +
response.availability().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
```

```
    try {
        GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
            .domainName(domainSuggestion)
            .suggestionCount(5)
            .onlyAvailable(true)
            .build();

        GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
        List<DomainSuggestion> suggestions = response.suggestionsList();
        for (DomainSuggestion suggestion : suggestions) {
            System.out.println("Suggestion Name: " + suggestion.domainName());
            System.out.println("Availability: " + suggestion.availability());
            System.out.println(" ");
        }

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
    try {
        ListPricesRequest pricesRequest = ListPricesRequest.builder()
            .tld(domainType)
            .build();

        ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
        listRes.stream()
            .flatMap(r -> r.prices().stream())
            .forEach(content -> System.out.println(" Name: " +
content.name() +
                " Registration: " + content.registrationPrice().price()
+ " "
                + content.registrationPrice().currency() +
                " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
    }
}
```

```
        System.exit(1);
    }
}

public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        LocalDateTime localDateTime2 = localDateTime.minusYears(1);
        Instant myStartTime = localDateTime2.toInstant(zoneOffset);
        Instant myEndTime = localDateTime.toInstant(zoneOffset);

        ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
            .start(myStartTime)
            .end(myEndTime)
            .build();

        ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
        listRes.stream()
            .flatMap(r -> r.billingRecords().stream())
            .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                " Operation: " + content.operationAsString() +
                " Price: " + content.price()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listOperations(Route53DomainsClient route53DomainsClient) {
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        localDateTime = localDateTime.minusYears(1);
        Instant myTime = localDateTime.toInstant(zoneOffset);
```

```

        ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
        .submittedSince(myTime)
        .build();

        ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
        listRes.stream()
        .flatMap(r -> r.operations().stream())
        .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                " Status: " + content.statusAsString() +
                " Date: " + content.submittedDate()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listDomains(Route53DomainsClient route53DomainsClient) {
    try {
        ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
        listRes.stream()
        .flatMap(r -> r.domains().stream())
        .forEach(content -> System.out.println("The domain name is " +
content.domainName()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}

```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [CheckDomainAvailability](#)
  - [CheckDomainTransferability](#)
  - [GetDomainDetail](#)
  - [GetDomainSuggestions](#)

- [GetOperationDetail](#)
- [ListDomains](#)
- [ListOperations](#)
- [ListPrices](#)
- [RegisterDomain](#)
- [ViewBilling](#)

## Contoh Amazon S3 menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan Amazon S3. AWS SDK for Java 2.x

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Memulai

Halo Amazon S3

Contoh kode berikut ini menunjukkan cara memulai menggunakan Amazon S3.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }

    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListBuckets](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

Tambahkan aturan CORS ke bucket

Contoh kode berikut menunjukkan cara menambahkan aturan cross-origin resource sharing (CORS) ke bucket S3.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import java.util.ArrayList;
import java.util.List;
import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3Cors {
    public static void main(String[] args) {
```

```
final String usage = ""

    Usage:
        <bucketName> <accountId>\s

    Where:
        bucketName - The Amazon S3 bucket to upload an object into.
        accountId - The id of the account that owns the Amazon S3
bucket.

    """;

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String bucketName = args[0];
String accountId = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

setCorsInformation(s3, bucketName, accountId);
getBucketCorsInformation(s3, bucketName, accountId);
deleteBucketCorsInformation(s3, bucketName, accountId);
s3.close();
}

public static void deleteBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
    try {
        DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
            .bucket(bucketName)
            .expectedBucketOwner(accountId)
            .build();

        s3.deleteBucketCors(bucketCorsRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



```
}

public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
    try {
        GetBucketCorsRequest bucketCorsRequest = GetBucketCorsRequest.builder()
            .bucket(bucketName)
            .expectedBucketOwner(accountId)
            .build();

        GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
        List<CORSRule> corsRules = corsResponse.corsRules();
        for (CORSRule rule : corsRules) {
            System.out.println("allowOrigins: " + rule.allowedOrigins());
            System.out.println("AllowedMethod: " + rule.allowedMethods());
        }

    } catch (S3Exception e) {

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
    List<String> allowMethods = new ArrayList<>();
    allowMethods.add("PUT");
    allowMethods.add("POST");
    allowMethods.add("DELETE");

    List<String> allowOrigins = new ArrayList<>();
    allowOrigins.add("http://example.com");
    try {
        // Define CORS rules.
        CORSRule corsRule = CORSRule.builder()
            .allowedMethods(allowMethods)
            .allowedOrigins(allowOrigins)
            .build();

        List<CORSRule> corsRules = new ArrayList<>();
        corsRules.add(corsRule);
        CORSConfiguration configuration = CORSConfiguration.builder()
```

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

    PutBucketCorsRequest putBucketCorsRequest =
    PutBucketCorsRequest.builder()
        .bucket(bucketName)
        .corsConfiguration(configuration)
        .expectedBucketOwner(accountId)
        .build();

    s3.putBucketCors(putBucketCorsRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [PutBucketCors](#) di Referensi AWS SDK for Java 2.x API.

Tambahkan konfigurasi siklus hidup ke bucket

Contoh kode berikut menunjukkan cara menambahkan konfigurasi siklus hidup ke bucket S3.

SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
```

```
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class LifecycleConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon Simple Storage Service
(Amazon S3) bucket to upload an object into.
                accountId - The id of the account that owns the
Amazon S3 bucket.

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
```

```
        .build();

        setLifecycleConfig(s3, bucketName, accountId);
        getLifecycleConfig(s3, bucketName, accountId);
        deleteLifecycleConfig(s3, bucketName, accountId);
        System.out.println("You have successfully created, updated, and
deleted a Lifecycle configuration");
        s3.close();
    }

    public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            // Create a rule to archive objects with the
"glacierobjects/" prefix to Amazon
            // S3 Glacier.
            LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

            Transition transition = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            LifecycleRule rule1 = LifecycleRule.builder()
                .id("Archive immediately rule")
                .filter(ruleFilter)
                .transitions(transition)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Create a second rule.
            Transition transition2 = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            List<Transition> transitionList = new ArrayList<>();
            transitionList.add(transition2);
```

```
        LifecycleRuleFilter ruleFilter2 =
LifecycleRuleFilter.builder()
                        .prefix("glacierobjects/")
                        .build();

        LifecycleRule rule2 = LifecycleRule.builder()
                .id("Archive and then delete rule")
                .filter(ruleFilter2)
                .transitions(transitionList)
                .status(ExpirationStatus.ENABLED)
                .build();

        // Add the LifecycleRule objects to an ArrayList.
        ArrayList<LifecycleRule> ruleList = new ArrayList<>();
        ruleList.add(rule1);
        ruleList.add(rule2);

        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                .rules(ruleList)
                .build();

        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)

                .lifecycleConfiguration(lifecycleConfiguration)
                .expectedBucketOwner(accountId)
                .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Retrieve the configuration and add a new rule.
    public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
```

```
GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
    .builder()
    .bucket(bucketName)
    .expectedBucketOwner(accountId)
    .build();

GetBucketLifecycleConfigurationResponse response = s3
    .getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
List<LifecycleRule> newList = new ArrayList<>();
List<LifecycleRule> rules = response.rules();
for (LifecycleRule rule : rules) {
    newList.add(rule);
}

// Add a new rule with both a prefix predicate and a tag
predicate.
LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
    .prefix("YearlyDocuments/")
    .build();

Transition transition = Transition.builder()
    .storageClass(TransitionStorageClass.GLACIER)
    .days(3650)
    .build();

LifecycleRule rule1 = LifecycleRule.builder()
    .id("NewRule")
    .filter(ruleFilter)
    .transitions(transition)
    .status(ExpirationStatus.ENABLED)
    .build();

// Add the new rule to the list.
newList.add(rule1);
BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
    .rules(newList)
    .build();
```

```
        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
        .builder()
        .bucket(bucketName)

        .lifecycleConfiguration(lifecycleConfiguration)
        .expectedBucketOwner(accountId)
        .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the configuration from the Amazon S3 bucket.
    public static void deleteLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
        try {
            DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
DeleteBucketLifecycleRequest

                .builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [PutBucketLifecycleConfiguration](#) di Referensi AWS SDK for Java 2.x API.

## Menambahkan kebijakan ke bucket

Contoh kode berikut menunjukkan cara menambahkan kebijakan ke bucket S3.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <polFile>

                Where:
                bucketName - The Amazon S3 bucket to set the policy on.
                polFile - A JSON file containing the policy (see the Amazon S3
                Readme for an example).\s
```



```
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String polFile = args[1];
    String policyText = getBucketPolicyFromFile(polFile);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setPolicy(s3, bucketName, policyText);
    s3.close();
}

public static void setPolicy(S3Client s3, String bucketName, String policyText)
{
    System.out.println("Setting policy:");
    System.out.println("----");
    System.out.println(policyText);
    System.out.println("----");
    System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

    try {
        PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
            .bucket(bucketName)
            .policy(policyText)
            .build();

        s3.putBucketPolicy(policyReq);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("Done!");
}

// Loads a JSON-formatted policy from a file
```

```
public static String getBucketPolicyFromFile(String policyFile) {

    StringBuilder fileText = new StringBuilder();
    try {
        List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
        for (String line : lines) {
            fileText.append(line);
        }

    } catch (IOException e) {
        System.out.format("Problem reading file: \"%s\"", policyFile);
        System.out.println(e.getMessage());
    }

    try {
        final JsonParser parser = new
ObjectMapper().getFactory().createParser(fileText.toString());
        while (parser.nextToken() != null) {
        }


    } catch (IOException jpe) {
        jpe.printStackTrace();
    }
    return fileText.toString();
}
}
```

- Untuk detail API, lihat [PutBucketPolicy](#) di Referensi AWS SDK for Java 2.x API.

Menyalin objek dari satu bucket ke bucket lainnya

Contoh kode berikut menunjukkan cara menyalin objek S3 dari satu ember ke ember lainnya.

SDK for Java 2.x

 Note

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

## Salin objek menggunakan [S3Client](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CopyObject {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <objectKey> <fromBucket> <toBucket>

            Where:
                objectKey - The name of the object (for example, book.pdf).
                fromBucket - The S3 bucket name that contains the object (for
example, bucket1).
                toBucket - The S3 bucket to copy the object to (for example,
bucket2).

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String objectKey = args[0];
        String fromBucket = args[1];
        String toBucket = args[2];
        System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
```

```

        .region(region)
        .build();

    copyBucketObject(s3, fromBucket, objectKey, toBucket);
    s3.close();
}

public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .sourceBucket(fromBucket)
        .sourceKey(objectKey)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        return copyRes.copyObjectResult().toString();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}

```

Gunakan [S3 TransferManager](#) untuk [menyalin objek](#) dari satu ember ke ember lainnya. Lihat [file lengkap](#) dan [lakukan pengujian](#).

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

```

```
public String copyObject(S3TransferManager transferManager, String bucketName,
    String key, String destinationBucket, String destinationKey) {
    CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
        .sourceBucket(bucketName)
        .sourceKey(key)
        .destinationBucket(destinationBucket)
        .destinationKey(destinationKey)
        .build();

    CopyRequest copyRequest = CopyRequest.builder()
        .copyObjectRequest(copyObjectRequest)
        .build();

    Copy copy = transferManager.copy(copyRequest);

    CompletedCopy completedCopy = copy.completionFuture().join();
    return completedCopy.response().copyObjectResult().eTag();
}
```

- Untuk detail API, lihat [CopyObject](#) di Referensi AWS SDK for Java 2.x API.

## Buat bucket

Contoh kode berikut menunjukkan cara membuat bucket S3.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

```
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateBucket {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The name of the bucket to create. The bucket name
must be unique, or an error occurs.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Creating a bucket named %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        createBucket(s3, bucketName);
        s3.close();
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
```

```

        .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [CreateBucket](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus kebijakan dari bucket

Contoh kode berikut menunjukkan cara menghapus kebijakan dari bucket S3.

### SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development

```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to delete the policy from (for
example, bucket1)."";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }

    // Delete the bucket policy.
    public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
        DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            s3.deleteBucketPolicy(delReq);
            System.out.println("Done!");
        }
    }
}
```



```
        } catch (S3Exception e) {  
            System.err.println(e.awsErrorDetails().errorMessage());  
            System.exit(1);  
        }  
    }  
}
```

- Untuk detail API, lihat [DeleteBucketPolicy](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus bucket kosong

Contoh kode berikut menunjukkan cara menghapus bucket S3 kosong.

### SDK for Java 2.x

#### Note

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

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()  
    .bucket(bucket)  
    .build();  
  
s3.deleteBucket(deleteBucketRequest);  
s3.close();
```

- Untuk detail API, lihat [DeleteBucket](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus beberapa objek

Contoh kode berikut menunjukkan cara menghapus beberapa objek dari bucket S3.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <bucketName>

                Where:
                    bucketName - the Amazon S3 bucket name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String bucketName = args[0];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

deleteBucketObjects(s3, bucketName);
s3.close();
}

public static void deleteBucketObjects(S3Client s3, String bucketName) {
    // Upload three sample objects to the specified Amazon S3 bucket.
    ArrayList<ObjectIdentifier> keys = new ArrayList<>();
    PutObjectRequest putOb;
    ObjectIdentifier objectId;

    for (int i = 0; i < 3; i++) {
        String keyName = "delete object example " + i;
        objectId = ObjectIdentifier.builder()
            .key(keyName)
            .build();

        putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        s3.putObject(putOb, RequestBody.fromString(keyName));
        keys.add(objectId);
    }

    System.out.println(keys.size() + " objects successfully created.");

    // Delete multiple objects in one request.
    Delete del = Delete.builder()
        .objects(keys)
        .build();

    try {
        DeleteObjectsRequest multiObjectDeleteRequest =
DeleteObjectsRequest.builder()
        .bucket(bucketName)
        .delete(del)
        .build();
```

```
s3.deleteObjects(multiObjectDeleteRequest);
System.out.println("Multiple objects are deleted!");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DeleteObjects](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus konfigurasi situs web dari bucket

Contoh kode berikut menunjukkan cara menghapus konfigurasi situs web dari bucket S3.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
```

```
final String usage = ""

        Usage:      <bucketName>

        Where:
            bucketName - The Amazon S3 bucket to delete the website
configuration from.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    System.out.format("Deleting website configuration for Amazon S3 bucket: %s
\n", bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    deleteBucketWebsiteConfig(s3, bucketName);
    System.out.println("Done!");
    s3.close();
}

public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName) {
    DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketWebsite(delReq);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.out.println("Failed to delete website configuration!");
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DeleteBucketWebsite](#) di Referensi AWS SDK for Java 2.x API.

## Menentukan keberadaan dan jenis konten suatu objek

Contoh kode berikut menunjukkan cara menentukan keberadaan dan jenis konten objek dalam bucket S3.

### SDK for Java 2.x

#### Note

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

Tentukan jenis konten suatu objek.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <keyName>>

                Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }

    public static void getContentType(S3Client s3, String bucketName, String
keyName) {
        try {
            HeadObjectRequest objectRequest = HeadObjectRequest.builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            HeadObjectResponse objectHead = s3.headObject(objectRequest);
            String type = objectHead.contentType();
            System.out.println("The object content type is " + type);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Dapatkan status pemulihan suatu objek.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <keyName>\s

                Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        checkStatus(s3, bucketName, keyName);
        s3.close();
    }

    public static void checkStatus(S3Client s3, String bucketName, String keyName) {
        try {
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            HeadObjectResponse response = s3.headObject(headObjectRequest);
            System.out.println("The Amazon S3 object restoration status is " +
                response.restore());
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```



```
    }  
  }  
}
```

- Untuk detail API, lihat [HeadObject](#) di Referensi AWS SDK for Java 2.x API.

## Mengunduh objek ke direktori lokal

Contoh Kode berikut ini menunjukkan cara mengunduh semua objek di bucket Amazon Simple Storage Service (Amazon S3) ke direktori lokal.

## SDK for Java 2.x

### Note

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

Gunakan [S3 TransferManager](#) untuk [mengunduh semua objek S3 di bucket](#) S3 yang sama. Lihat [file lengkap](#) dan [lakukan pengujian](#).

```
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.core.sync.RequestBody;  
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;  
import software.amazon.awssdk.transfer.s3.S3TransferManager;  
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;  
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;  
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;  
  
import java.io.IOException;  
import java.net.URI;  
import java.net.URISyntaxException;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.util.HashSet;  
import java.util.Set;  
import java.util.UUID;  
import java.util.stream.Collectors;
```

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
    .destination(Paths.get(destinationPathURI))
    .bucket(bucketName)
    .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}
```

- Untuk detail API, lihat [DownloadDirectory](#) di Referensi AWS SDK for Java 2.x API.

## Mengaktifkan notifikasi

Contoh kode berikut menunjukkan cara mengaktifkan notifikasi untuk bucket S3.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Event;
import software.amazon.awssdk.services.s3.model.NotificationConfiguration;
import
    software.amazon.awssdk.services.s3.model.PutBucketNotificationConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.TopicConfiguration;
import java.util.ArrayList;
import java.util.List;
```

```
public class SetBucketEventBridgeNotification {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket.\s
                topicArn - The Simple Notification Service topic ARN.\s
                id - An id value used for the topic configuration. This value is
displayed in the AWS Management Console.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String topicArn = args[1];
        String id = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3Client = S3Client.builder()
            .region(region)
            .build();

        setBucketNotification(s3Client, bucketName, topicArn, id);
        s3Client.close();
    }

    public static void setBucketNotification(S3Client s3Client, String bucketName,
String topicArn, String id) {
        try {
            List<Event> events = new ArrayList<>();
            events.add(Event.S3_OBJECT_CREATED_PUT);

            TopicConfiguration config = TopicConfiguration.builder()
                .topicArn(topicArn)
                .events(events)
                .id(id)
                .build();
```

```
List<TopicConfiguration> topics = new ArrayList<>();
topics.add(config);

NotificationConfiguration configuration =
NotificationConfiguration.builder()
    .topicConfigurations(topics)
    .build();

PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
    .builder()
    .bucket(bucketName)
    .notificationConfiguration(configuration)
    .skipDestinationValidation(true)
    .build();

// Set the bucket notification configuration.
s3Client.putBucketNotificationConfiguration(configurationRequest);
System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Untuk detail API, lihat [PutBucketNotificationConfiguration](#) di Referensi AWS SDK for Java 2.x API.

## Mendapatkan objek dari bucket

Contoh kode berikut menunjukkan cara membaca data dari objek dalam bucket S3.

## SDK for Java 2.x

 Note

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

Membaca data sebagai array byte menggunakan [S3Client](#).

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s

            """;
```

```
        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getObjectBytes(s3, bucketName, keyName, path);
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Gunakan [S3 TransferManager](#) untuk [mengunduh objek](#) dalam bucket S3 ke file lokal. Lihat [file lengkap](#) dan [lakukan pengujian](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String bucketName,
                            String key, String downloadedFilePath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
            .getObjectRequest(b -> b.bucket(bucketName).key(key))
            .addTransferListener(LoggingTransferListener.create())
            .destination(Paths.get(downloadedFilePath))
            .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
```

Baca tanda milik objek menggunakan [S3Client](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listTags(s3, bucketName, keyName);
        s3.close();
    }

    public static void listTags(S3Client s3, String bucketName, String keyName) {
```



```
    try {
        GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        GetObjectTaggingResponse tags = s3.getObjectTagging(getTaggingRequest);
        List<Tag> tagSet = tags.tagSet();
        for (Tag tag : tagSet) {
            System.out.println(tag.key());
            System.out.println(tag.value());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Dapatkan URL untuk objek menggunakan [S3Client](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectUrl {
    public static void main(String[] args) {
        final String usage = ""
```

```

Usage:
    <bucketName> <keyName>\s

Where:
    bucketName - The Amazon S3 bucket name.
    keyName - A key name that represents the object.\s
""";

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String bucketName = args[0];
String keyName = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

getUrl(s3, bucketName, keyName);
s3.close();
}

public static void getUrl(S3Client s3, String bucketName, String keyName) {
    try {
        GetUrlRequest request = GetUrlRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        URL url = s3.utilities().getUrl(request);
        System.out.println("The URL for " + keyName + " is " + url);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

Dapatkan objek dengan menggunakan objek klien S3Presigner menggunakan [S3Client](#).

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents a text file.\s
            """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
            .region(region)
```

```

        .build();

        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(60))
                .getObjectRequest(getObjectRequest)
                .build();

            PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
            String theUrl = presignedGetObjectRequest.url().toString();
            System.out.println("Presigned URL: " + theUrl);
            HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
            presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
                values.forEach(value -> {
                    connection.setRequestProperty(header, value);
                });
            });

            // Send any request payload that the service needs (not needed when
// isBrowserExecutable is true).
            if (presignedGetObjectRequest.signedPayload().isPresent()) {
                connection.setDoOutput(true);

                try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
                    OutputStream httpOutputStream =
connection.getOutputStream()) {
                    IoUtils.copy(signedPayload, httpOutputStream);
                }
            }
        }
    }

```

```

    }

    // Download the result of executing the request.
    try (InputStream content = connection.getInputStream()) {
        System.out.println("Service returned response: ");
        IoUtils.copy(content, System.out);
    }

} catch (S3Exception | IOException e) {
    e.printStackTrace();
}
}
}
}

```

Dapatkan objek dengan menggunakan ResponseTransformer objek dan [S3Client](#).

```

import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetDataResponseTransformer {
    public static void main(String[] args) {
        final String usage = ""

```

Usage:

```
        <bucketName> <keyName> <path>

        Where:
            bucketName - The Amazon S3 bucket name.\s
            keyName - The key name.\s
            path - The path where the file is written to.\s
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String keyName = args[1];
    String path = args[2];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    getObjectBytes(s3, bucketName, keyName, path);
    s3.close();
}

public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3.getObject(objectRequest, ResponseTransformer.toBytes());
        byte[] data = objectBytes.asByteArray();

        // Write the data to a local file.
        File myFile = new File(path);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
        System.out.println("Successfully obtained bytes from an S3 object");
        os.close();
    }
}
```

```
        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [GetObject](#) di Referensi AWS SDK for Java 2.x API.

## Mendapatkan ACL dari bucket

Contoh kode berikut menunjukkan cara mendapatkan daftar kontrol akses (ACL) dari bucket S3.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```

public class GetAcl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <objectKey>

            Where:
                bucketName - The Amazon S3 bucket to get the access control list
(ACL) for.
                objectKey - The object to get the ACL for.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    public static String getBucketACL(S3Client s3, String objectKey, String
bucketName) {
        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
            List<Grant> grants = aclRes.grants();
            String grantee = "";
            for (Grant grant : grants) {

```



```

        System.out.format("  %s: %s\n", grant.grantee().id(),
grant.permission());
        grantee = grant.grantee().id();
    }

    return grantee;
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return "";
}
}

```

- Untuk detail API, lihat [GetBucketAcl](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan kebijakan untuk bucket

Contoh kode berikut menunjukkan cara mendapatkan kebijakan untuk bucket S3.

SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */

```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to get the policy from.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }

    public static String getPolicy(S3Client s3, String bucketName) {
        String policyText;
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
            policyText = policyRes.policy();
            return policyText;
        } catch (S3Exception e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
}
```

- Untuk detail API, lihat [GetBucketPolicy](#) di Referensi AWS SDK for Java 2.x API.

## Buat daftar bucket

Contoh kode berikut menunjukkan cara membuat daftar bucket S3.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
```

```
        .build();

        listAllBuckets(s3);

    }


    public static void listAllBuckets(S3Client s3) {
        ListBucketsResponse response = s3.listBuckets();
        List<Bucket> bucketList = response.buckets();
        for (Bucket bucket: bucketList) {
            System.out.println("Bucket name "+bucket.name());
        }
    }
}
```

- Untuk detail API, lihat [ListBuckets](#) di Referensi AWS SDK for Java 2.x API.

Membuat daftar unggahan multibagian yang sedang berlangsung

Contoh kode berikut menunjukkan cara mendaftar unggahan multibagian yang sedang berlangsung pada sebuah bucket S3.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
        listUploads(s3, bucketName);
        s3.close();
    }

    public static void listUploads(S3Client s3, String bucketName) {
        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();

            ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
            List<MultipartUpload> uploads = response.uploads();
            for (MultipartUpload upload : uploads) {
                System.out.println("Upload in progress: Key = \"" + upload.key() +
"\", id = " + upload.uploadId());
            }
        }
    }
}
```

```
    }  
  
    } catch (S3Exception e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}  
}
```

- Untuk detail API, lihat [ListMultipartUploads](#) di Referensi AWS SDK for Java 2.x API.

## Membuat daftar objek dalam bucket

Contoh kode berikut menunjukkan cara membuat daftar objek dalam bucket S3.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;  
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.services.s3.model.S3Object;  
import java.util.List;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
  
public class ListObjects {
```

```
public static void main(String[] args) {
    final String usage = ""

        Usage:
            <bucketName>\s

        Where:
            bucketName - The Amazon S3 bucket from which objects are read.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    listBucketObjects(s3, bucketName);
    s3.close();
}

public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.print("\n The name of the key is " + myValue.key());
            System.out.print("\n The object is " + calKb(myValue.size()) + "
KBs");

            System.out.print("\n The owner is " + myValue.owner());
        }
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```

    }
}

// convert bytes to kbs.
private static long calcKb(Long val) {
    return val / 1024;
}
}

```

Daftarkan objek menggunakan paginasi.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName>\s

                Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }
}

```



```
public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsV2Request listReq = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
        listRes.stream()
            .flatMap(r -> r.contents().stream())
            .forEach(content -> System.out.println(" Key: " + content.key()
+ " size = " + content.size()));


    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ListObjectsV2](#) di Referensi AWS SDK for Java 2.x API.

Mengembalikan salinan objek yang diarsipkan

Contoh kode berikut menunjukkan cara mengembalikan salinan objek yang diarsipkan kembali ke bucket S3.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;

/*
 * For more information about restoring an object, see "Restoring an archived
 object" at
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <expectedBucketOwner>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name of an object with a Storage class value
of Glacier.\s
                expectedBucketOwner - The account that owns the bucket (you can
obtain this value from the AWS Management Console).\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
    }
}
```

```
s3.close();
}

public static void restoreS3Object(S3Client s3, String bucketName, String
keyName, String expectedBucketOwner) {
    try {
        RestoreRequest restoreRequest = RestoreRequest.builder()
            .days(10)

.glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
            .build();

        RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
            .expectedBucketOwner(expectedBucketOwner)
            .bucket(bucketName)
            .key(keyName)
            .restoreRequest(restoreRequest)
            .build();

        s3.restoreObject(objectRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [RestoreObject](#) di Referensi AWS SDK for Java 2.x API.

## Menetapkan ACL baru untuk bucket

Contoh kode berikut menunjukkan cara mengatur daftar kontrol akses (ACL) baru untuk bucket S3.

SDK for Java 2.x

### Note

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

```
import java.util.ArrayList;
import java.util.List;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Type;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <bucketName> <id>\s

            Where:
            bucketName - The Amazon S3 bucket to grant permissions on.\s
            id - The ID of the owner of this bucket (you can get this value
from the AWS Management Console).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
```

```
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

setBucketAcl(s3, bucketName, id);
System.out.println("Done!");
s3.close();
}

public static void setBucketAcl(S3Client s3, String bucketName, String id) {
    try {
        Grant ownerGrant = Grant.builder()
            .grantee(builder -> builder.id(id)
                .type(Type.CANONICAL_USER))
            .permission(Permission.FULL_CONTROL)
            .build();

        List<Grant> grantList2 = new ArrayList<>();
        grantList2.add(ownerGrant);

        AccessControlPolicy acl = AccessControlPolicy.builder()
            .owner(builder -> builder.id(id))
            .grants(grantList2)
            .build();

        PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
            .bucket(bucketName)
            .accessControlPolicy(acl)
            .build();

        s3.putBucketAcl(putAclReq);

    } catch (S3Exception e) {
        e.printStackTrace();
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [PutBucketAcl](#) di Referensi AWS SDK for Java 2.x API.

## Menetapkan konfigurasi situs web untuk bucket

Contoh kode berikut menunjukkan cara mengatur konfigurasi situs web untuk bucket S3.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <bucketName> [indexdoc]\s

                Where:
                bucketName  - The Amazon S3 bucket to set the website
configuration on.\s
                indexdoc - The index document, ex. 'index.html'
                        If not specified, 'index.html' will be set.

                """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String bucketName = args[0];
    String indexDoc = "index.html";
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setWebsiteConfig(s3, bucketName, indexDoc);
    s3.close();
}

public static void setWebsiteConfig(S3Client s3, String bucketName, String
indexDoc) {
    try {
        WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
            .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
            .build();

        PutBucketWebsiteRequest pubWebsiteReq =
PutBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .websiteConfiguration(websiteConfig)
            .build();

        s3.putBucketWebsite(pubWebsiteReq);
        System.out.println("The call was successful");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [PutBucketWebsite](#) di Referensi AWS SDK for Java 2.x API.

## Mengunggah objek ke bucket

Contoh kode berikut menunjukkan cara mengunggah objek ke bucket S3.

## SDK for Java 2.x

 Note

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

Unggah file ke bucket menggunakan [S3Client](#).

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutObject {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <objectKey> <objectPath>\s

                Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                objectKey - The object to upload (for example, book.pdf).
                objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
                """;

        if (args.length != 3) {
```



```
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String objectKey = args[1];
    String objectPath = args[2];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    putS3Object(s3, bucketName, objectKey, objectPath);
    s3.close();
}

// This example uses RequestBody.fromFile to avoid loading the whole file into
// memory.
public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
    try {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("x-amz-meta-myVal", "test");
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .metadata(metadata)
            .build();

        s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
        System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

Gunakan [S3 TransferManager](#) untuk [mengunggah file](#) ke bucket. Lihat [file lengkap](#) dan [lakukan pengujian](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

public String uploadFile(S3TransferManager transferManager, String bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create())
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
    return uploadResult.response().eTag();
}
```

Unggah objek ke bucket dan tetapkan tanda menggunakan [S3Client](#).

```
public static void putS3ObjectTags(S3Client s3, String bucketName, String
objectKey, String objectPath) {
    try {
        Tag tag1 = Tag.builder()
            .key("Tag 1")
            .value("This is tag 1")
            .build();

        Tag tag2 = Tag.builder()
            .key("Tag 2")
            .value("This is tag 2")
            .build();
    }
```

```
List<Tag> tags = new ArrayList<>();
tags.add(tag1);
tags.add(tag2);

Tagging allTags = Tagging.builder()
    .tagSet(tags)
    .build();

PutObjectRequest putOb = PutObjectRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .tagging(allTags)
    .build();

s3.putObject(putOb, RequestBody.fromBytes(getObjectFile(objectPath)));

} catch (S3Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
    try {
        GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
        List<Tag> obTags = getTaggingRes.tagSet();
        for (Tag sinTag : obTags) {
            System.out.println("The tag key is: " + sinTag.key());
            System.out.println("The tag value is: " + sinTag.value());
        }

        // Replace the object's tags with two new tags.
        Tag tag3 = Tag.builder()
            .key("Tag 3")
            .value("This is tag 3")
            .build();
```

```
        Tag tag4 = Tag.builder()
            .key("Tag 4")
            .value("This is tag 4")
            .build();

        List<Tag> tags = new ArrayList<>();
        tags.add(tag3);
        tags.add(tag4);

        Tagging updatedTags = Tagging.builder()
            .tagSet(tags)
            .build();

        PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .tagging(updatedTags)
            .build();

        s3.putObjectTagging(taggingRequest1);
        GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
        List<Tag> modTags = getTaggingRes2.tagSet();
        for (Tag sinTag : modTags) {
            System.out.println("The tag key is: " + sinTag.key());
            System.out.println("The tag value is: " + sinTag.value());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Return a byte array.
private static byte[] getObjectFile(String filePath) {
    FileInputStream fileInputStream = null;
    byte[] byteArray = null;

    try {
        File file = new File(filePath);
        byteArray = new byte[(int) file.length()];
```

```
        fileInputStream = new FileInputStream(file);
        fileInputStream.read(bytesArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    return bytesArray;
}
}
```

Unggah objek ke bucket dan tetapkan metadata menggunakan [S3Client](#).

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = ""
```

**Usage:**

```
<bucketName> <objectKey> <objectPath>\s
```

**Where:**

bucketName - The Amazon S3 bucket to upload an object into.

objectKey - The object to upload (for example, book.pdf).

objectPath - The path where the file is located (for example, C:/

AWS/book2.pdf).\s

```
""";
```

```
if (args.length != 3) {
    System.out.println(USAGE);
    System.exit(1);
}
```

```
String bucketName = args[0];
String objectKey = args[1];
String objectPath = args[2];
System.out.println("Putting object " + objectKey + " into bucket " +
bucketName);
```

```
System.out.println(" in bucket: " + bucketName);
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

```
putS3Object(s3, bucketName, objectKey, objectPath);
s3.close();
```

```
}
```

```
// This example uses RequestBody.fromFile to avoid loading the whole file into
// memory.
```

```
public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
```

```
    try {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("author", "Mary Doe");
        metadata.put("version", "1.0.0.0");
```

```
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .metadata(metadata)
            .build();
```

```

        s3.putObject(putObj, RequestBody.fromFile(new File(objectPath)));
        System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}

```

Unggah objek ke bucket dan tetapkan nilai retensi objek menggunakan [S3Client](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutObjectRetention {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <key> <bucketName>\s

                Where:
                key - The name of the object (for example, book.pdf).\s

```

```

        bucketName - The Amazon S3 bucket name that contains the object
(for example, bucket1).\s
        """";

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String key = args[0];
    String bucketName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setRetentionPeriod(s3, key, bucketName);
    s3.close();
}

public static void setRetentionPeriod(S3Client s3, String key, String bucket) {
    try {
        LocalDate localDate = LocalDate.parse("2020-07-17");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        ObjectLockRetention lockRetention = ObjectLockRetention.builder()
            .mode("COMPLIANCE")
            .retainUntilDate(instant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucket)
            .key(key)
            .bypassGovernanceRetention(true)
            .retention(lockRetention)
            .build();

        // To set Retention on an object, the Amazon S3 bucket must support
object
        // locking, otherwise an exception is thrown.
        s3.putObjectRetention(retentionRequest);
    }
}

```



```
        System.out.print("An object retention configuration was successfully
placed on the object");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [PutObject](#) di Referensi AWS SDK for Java 2.x API.

## Mengunggah direktori ke bucket

Contoh kode berikut ini menunjukkan cara mengunggah direktori lokal secara rekursif ke bucket Amazon Simple Storage Service (Amazon S3).

### SDK for Java 2.x

#### Note

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

Gunakan [S3 TransferManager](#) untuk [mengunggah direktori lokal](#). Lihat [file lengkap](#) dan [lakukan pengujian](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;
```

```
public Integer uploadDirectory(S3TransferManager transferManager,
    URI sourceDirectory, String bucketName) {
    DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
    .source(Paths.get(sourceDirectory))
    .bucket(bucketName)
    .build());

    CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
    completedDirectoryUpload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryUpload.failedTransfers().size();
}
```

- Untuk detail API, lihat [UploadDirectory](#) di Referensi AWS SDK for Java 2.x API.

## Gunakan SQL dengan Amazon S3 Select

Contoh kode berikut menunjukkan bagaimana untuk mengambil subset data menggunakan SQL.

SDK for Java 2.x

### Note

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

Contoh berikut menunjukkan query menggunakan objek JSON. [Contoh lengkap](#) juga menunjukkan penggunaan objek CSV.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
```

```
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
    LoggerFactory.getLogger(SelectObjectContentExample.class);
    static final String BUCKET_NAME = "select-object-content-" + UUID.randomUUID();
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
        SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
```

```
        logger.error(e.getMessage(), e);
        System.exit(1);
    } finally {
        SelectObjectContentExample.tearDown();
    }
}

EventStreamInfo runSelectObjectContentMethodForJSON() {
    // Set up request parameters.
    final String queryExpression = "select * from s3object[*][*] c where c.area
< 350000";
    final String fileType = FILE_JSON;

    InputSerialization inputSerialization = InputSerialization.builder()
        .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
        .compressionType(CompressionType.NONE)
        .build();

    OutputSerialization outputSerialization = OutputSerialization.builder()
        .json(JSONOutput.builder().recordDelimiter(null).build())
        .build();

    // Build the SelectObjectContentRequest.
    SelectObjectContentRequest select = SelectObjectContentRequest.builder()
        .bucket(BUCKET_NAME)
        .key(FILE_JSON)
        .expression(queryExpression)
        .expressionType(ExpressionType.SQL)
        .inputSerialization(inputSerialization)
        .outputSerialization(outputSerialization)
        .build();

    EventStreamInfo eventStreamInfo = new EventStreamInfo();
    // Call the selectObjectContent method with the request and a response
handler.
    // Supply an EventStreamInfo object to the response handler to gather
records and information from the response.
    s3AsyncClient.selectObjectContent(select,
buildResponseHandler(eventStreamInfo)).join();

    // Log out information gathered while processing the response stream.
    long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record ->
        record.split("\n").length
    ).sum();
}
```

```

        logger.info("Total records {}: {}", fileType, recordCount);
        logger.info("Visitor onRecords for fileType {} called {} times", fileType,
eventStreamInfo.getCountOnRecordsCalled());
        logger.info("Visitor onStats for fileType {}, {}", fileType,
eventStreamInfo.getStats());
        logger.info("Visitor onContinuations for fileType {}, {}", fileType,
eventStreamInfo.getCountContinuationEvents());
        return eventStreamInfo;
    }

    static SelectObjectContentResponseHandler buildResponseHandler(EventStreamInfo
eventStreamInfo) {
        // Use a Visitor to process the response stream. This visitor logs
information and gathers details while processing.
        final SelectObjectContentResponseHandler.Visitor visitor =
SelectObjectContentResponseHandler.Visitor.builder()
            .onRecords(r -> {
                logger.info("Record event received.");
                eventStreamInfo.addRecord(r.payload().asUtf8String());
                eventStreamInfo.incrementOnRecordsCalled();
            })
            .onCont(ce -> {
                logger.info("Continuation event received.");
                eventStreamInfo.incrementContinuationEvents();
            })
            .onProgress(pe -> {
                Progress progress = pe.details();
                logger.info("Progress event received:\n bytesScanned:
{} \n bytesProcessed: {} \n bytesReturned: {}",
                    progress.bytesScanned(),
                    progress.bytesProcessed(),
                    progress.bytesReturned());
            })
            .onEnd(ee -> logger.info("End event received."))
            .onStats(se -> {
                logger.info("Stats event received.");
                eventStreamInfo.addStats(se.details());
            })
            .build();

        // Build the SelectObjectContentResponseHandler with the visitor that
processes the stream.
        return SelectObjectContentResponseHandler.builder()
            .subscriber(visitor).build();
    }

```

```
}

// The EventStreamInfo class is used to store information gathered while
processing the response stream.
static class EventStreamInfo {
    private final List<String> records = new ArrayList<>();
    private Integer countOnRecordsCalled = 0;
    private Integer countContinuationEvents = 0;
    private Stats stats;

    void incrementOnRecordsCalled() {
        countOnRecordsCalled++;
    }

    void incrementContinuationEvents() {
        countContinuationEvents++;
    }

    void addRecord(String record) {
        records.add(record);
    }

    void addStats(Stats stats) {
        this.stats = stats;
    }

    public List<String> getRecords() {
        return records;
    }

    public Integer getCountOnRecordsCalled() {
        return countOnRecordsCalled;
    }

    public Integer getCountContinuationEvents() {
        return countContinuationEvents;
    }

    public Stats getStats() {
        return stats;
    }
}
```

- Untuk detail API, lihat [SelectObjectContent](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

Membuat URL yang telah ditetapkan sebelumnya

Contoh kode berikut menunjukkan cara membuat URL presigned untuk Amazon S3 dan mengunggah objek.

SDK for Java 2.x

### Note

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

Buat URL yang telah ditandatangani sebelumnya untuk suatu objek, lalu unduh (GET request).

Impor.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
```

```
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

## Hasilkan URL.

```
/* Create a pre-signed URL to download an object in a subsequent GET request. */
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest presignRequest =
            GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10)) // The URL will
                expire in 10 minutes.
                .getObjectRequest(objectRequest)
                .build();

        PresignedGetObjectRequest presignedRequest =
            presigner.presignGetObject(presignRequest);
        logger.info("Presigned URL: [{}]", presignedRequest.url().toString());
        logger.info("HTTP method: [{}]",
            presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}
```

Unduh objek dengan menggunakan salah satu dari tiga pendekatan berikut.

Gunakan kelas JDK `HttpURLConnection` (sejak v1.1) untuk melakukan download.



```
/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setRequestMethod("GET");
        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());
    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}
```

Gunakan kelas JDK `HttpClient` (sejak v11) untuk melakukan download.

```
/* Use the JDK HttpClient (since v11) class to do the download. */
public byte[] useHttpClientToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpResponse<InputStream> response = httpClient.send(requestBuilder
            .uri(presignedUrl.toURI())
            .GET()
            .build(),
            HttpResponse.BodyHandlers.ofInputStream());

        IoUtils.copy(response.body(), byteArrayOutputStream);

        logger.info("HTTP response code is " + response.statusCode());
    }
```

```

    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

Gunakan kelas AWS SDK for SdkHttpClient Java untuk melakukan download.

```

/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToPut(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            response.responseBody().ifPresentOrElse(
                abortableInputStream -> {
                    try {
                        IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                },
                () -> logger.error("No response body."));

            logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
        }
    }
}

```

```
    } catch (URISyntaxException | IOException e) {  
        logger.error(e.getMessage(), e);  
    }  
    return byteArrayOutputStream.toByteArray();  
}
```

Buat URL yang telah ditandatangani sebelumnya untuk unggahan, lalu unggah file (permintaan PUT).

Impor.

```
import com.example.s3.util.PresignUrlUtils;  
import org.slf4j.Logger;  
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;  
import software.amazon.awssdk.http.HttpExecuteRequest;  
import software.amazon.awssdk.http.HttpExecuteResponse;  
import software.amazon.awssdk.http.SdkHttpClient;  
import software.amazon.awssdk.http.SdkHttpMethod;  
import software.amazon.awssdk.http.SdkHttpRequest;  
import software.amazon.awssdk.http.apache.ApacheHttpClient;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.PutObjectRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.services.s3.presigner.S3Presigner;  
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;  
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;  
  
import java.io.File;  
import java.io.IOException;  
import java.io.OutputStream;  
import java.io.RandomAccessFile;  
import java.net.HttpURLConnection;  
import java.net.URISyntaxException;  
import java.net.URL;  
import java.net.http.HttpClient;  
import java.net.http.HttpRequest;  
import java.net.http.HttpResponse;  
import java.nio.ByteBuffer;  
import java.nio.channels.FileChannel;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.time.Duration;
```

```
import java.util.Map;
import java.util.UUID;
```

## Hasilkan URL.

```
/* Create a presigned URL to use in a subsequent PUT request */
public String createPresignedUrl(String bucketName, String keyName, Map<String,
String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .metadata(metadata)
            .build();

        PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL expires
in 10 minutes.
            .putObjectRequest(objectRequest)
            .build();

        PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload a file to: [{}]", myURL);
        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}
```

Unggah objek file dengan menggunakan salah satu dari tiga pendekatan berikut.

Gunakan kelas JDK `URLConnection` (sejak v1.1) untuk melakukan upload.

```
/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
```

```

    public void useHttpURLConnectionToPut(String presignedUrlString, File fileToPut,
    Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
            connection.setDoOutput(true);
            metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" +
k, v));
            connection.setRequestMethod("PUT");
            OutputStream out = connection.getOutputStream();

            try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
                FileChannel inChannel = file.getChannel()) {
                ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

                while (inChannel.read(buffer) > 0) {
                    buffer.flip();
                    for (int i = 0; i < buffer.limit(); i++) {
                        out.write(buffer.get());
                    }
                    buffer.clear();
                }
            } catch (IOException e) {
                logger.error(e.getMessage(), e);
            }

            out.close();
            connection.getResponseCode();
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
}

```

Gunakan kelas JDK `HttpClient` (sejak v11) untuk melakukan upload.

```

/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {

```

```

logger.info("Begin [{}] upload", fileToPut.toString());

HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

HttpClient httpClient = HttpClient.newHttpClient();
try {
    final HttpResponse<Void> response = httpClient.send(requestBuilder
        .uri(new URL(presignedUrlString).toURI())
        .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI()))
            .build(),
            HttpResponse.BodyHandlers.discarding());

    logger.info("HTTP response code is " + response.statusCode());

} catch (URISyntaxException | InterruptedException | IOException e) {
    logger.error(e.getMessage(), e);
}
}

```

Gunakan `SdkHttpClient` kelas AWS for Java V2 untuk melakukan upload.

```

/* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());
        // Add headers
        metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" + k,
v));

        // Finish building the request.
        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)

```

```
        .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
        .build();

    try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
        HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
        logger.info("Response code: {}",
response.httpResponse().statusCode());
    }
} catch (URISyntaxException | IOException e) {
    logger.error(e.getMessage(), e);
}
}
```

## Memulai bucket dan objek

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- 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 for Java 2.x

### Note

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

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This Java code example performs the following tasks:
*
* 1. Creates an Amazon S3 bucket.
* 2. Uploads an object to the bucket.
* 3. Downloads the object to another local file.
* 4. Uploads an object using multipart upload.
* 5. List all objects located in the Amazon S3 bucket.
* 6. Copies the object to another Amazon S3 bucket.
* 7. Deletes the object from the Amazon S3 bucket.
* 8. Deletes the Amazon S3 bucket.
*/
```

```
public class S3Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <bucketName> <key> <objectPath> <savePath> <toBucket>

            Where:
                bucketName - The Amazon S3 bucket to create.
                key - The key to use.
                objectPath - The path where the file is located (for example,
C:/AWS/book2.pdf).
                savePath - The path where the file is saved after it's
downloaded (for example, C:/AWS/book2.pdf).
                toBucket - An Amazon S3 bucket to where an object is copied to
(for example, C:/AWS/book2.pdf).\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
```



```
String toBucket = args[4];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon S3 example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an Amazon S3 bucket.");
createBucket(s3, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Update a local file to the Amazon S3 bucket.");
uploadLocalFile(s3, bucketName, key, objectPath);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Download the object to another local file.");
getObjectBytes(s3, bucketName, key, savePath);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Perform a multipart upload.");
String multipartKey = "multiPartKey";
multipartUpload(s3, toBucket, multipartKey);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. List all objects located in the Amazon S3 bucket.");
listAllObjects(s3, bucketName);
anotherListExample(s3, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Copy the object to another Amazon S3 bucket.");
copyBucketObject(s3, bucketName, key, toBucket);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the object from the Amazon S3 bucket.");
```

```
deleteObjectFromBucket(s3, bucketName, key);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Delete the Amazon S3 bucket.");
deleteBucket(s3, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("All Amazon S3 operations were successfully performed");
System.out.println(DASHES);
s3.close();
}

// Create a bucket by using a S3Waiter object.
public static void createBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteBucket(S3Client client, String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();
```

```
        client.deleteBucket(deleteBucketRequest);
        System.out.println(bucket + " was deleted.");
    }

    /**
     * Upload an object in parts.
     */
    public static void multipartUpload(S3Client s3, String bucketName, String key) {
        int mB = 1024 * 1024;
        // First create a multipart upload and get the upload id.
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = response.uploadId();
        System.out.println(uploadId);

        // Upload all the different parts of the object.
        UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .partNumber(1).build();

        String etag1 = s3.uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
            .eTag();
        CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
            .uploadId(uploadId)
            .partNumber(2).build();
        String etag2 = s3.uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB)))
            .eTag();
        CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();
```

```
        // Call completeMultipartUpload operation to tell S3 to merge all uploaded
        // parts and finish the multipart operation.
        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
            .parts(part1, part2)
            .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .multipartUpload(completedMultipartUpload)
            .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
    }

    private static ByteBuffer getRandomByteBuffer(int size) {
        byte[] b = new byte[size];
        new Random().nextBytes(b);
        return ByteBuffer.wrap(b);
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();
        }
    }
}
```

```
    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void uploadLocalFile(S3Client s3, String bucketName, String key,
String objectPath) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    s3.putObject(objectRequest, RequestBody.fromFile(new File(objectPath)));
}

public static void listAllObjects(S3Client s3, String bucketName) {
    ListObjectsV2Request listObjectsReqManual = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    boolean done = false;
    while (!done) {
        ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
        for (S3Object content : listObjResponse.contents()) {
            System.out.println(content.key());
        }

        if (listObjResponse.nextContinuationToken() == null) {
            done = true;
        }

        listObjectsReqManual = listObjectsReqManual.toBuilder()
            .continuationToken(listObjResponse.nextContinuationToken())
            .build();
    }
}

public static void anotherListExample(S3Client s3, String bucketName) {
    ListObjectsV2Request listReq = ListObjectsV2Request.builder()
```

```
        .bucket(bucketName)
        .maxKeys(1)
        .build();

ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);

// Process response pages.
listRes.stream()
    .flatMap(r -> r.contents().stream())
    .forEach(content -> System.out.println(" Key: " + content.key() + "
size = " + content.size()));

// Helper method to work with paginated collection of items directly.
listRes.contents().stream()
    .forEach(content -> System.out.println(" Key: " + content.key() + "
size = " + content.size()));

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

public static void deleteObjectFromBucket(S3Client s3, String bucketName, String
key) {
    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    s3.deleteObject(deleteObjectRequest);
    System.out.println(key + " was deleted");
}

public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
    String encodedUrl = null;
    try {
        encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
StandardCharsets.UTF_8.toString());
    } catch (UnsupportedEncodingException e) {
        System.out.println("URL could not be encoded: " + e.getMessage());
    }
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
```

```
        .copySource(encodedUrl)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        System.out.println("The " + objectKey + " was copied to " + toBucket);
        return copyRes.copyObjectResult().toString();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjectsV2](#)
  - [PutObject](#)

## Mengurai URI

Contoh kode berikut menunjukkan cara mengurai URI Amazon S3 untuk mengekstrak komponen penting seperti nama bucket dan kunci objek.

## SDK for Java 2.x

 Note

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

Mengurai Amazon S3 URI dengan menggunakan kelas [S3Uri](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

/**
 *
 * @param s3Client - An S3Client through which you acquire an S3Uri instance.
 * @param s3objectUrl - A complex URL (String) that is used to demonstrate S3Uri
 * capabilities.
 */
public static void parseS3UriExample(S3Client s3Client, String s3objectUrl) {
    logger.info(s3objectUrl);
    // Console output:
    // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.

    // Create an S3Utilities object using the configuration of the s3Client.
    S3Utilities s3Utilities = s3Client.utilities();

    // From a String URL create a URI object to pass to the parseUri() method.
    URI uri = URI.create(s3objectUrl);
    S3Uri s3Uri = s3Utilities.parseUri(uri);

    // If the URI contains no value for the Region, bucket or key, the SDK
returns
    // an empty Optional.
```



```
// The SDK returns decoded URI values.

Region region = s3Uri.region().orElse(null);
log("region", region);
// Console output: 'region: us-west-1'.

String bucket = s3Uri.bucket().orElse(null);
log("bucket", bucket);
// Console output: 'bucket: myBucket'.

String key = s3Uri.key().orElse(null);
log("key", key);
// Console output: 'key: resources/doc.txt'.

Boolean isPathStyle = s3Uri.isPathStyle();
log("isPathStyle", isPathStyle);
// Console output: 'isPathStyle: true'.

// If the URI contains no query parameters, the SDK returns an empty map.
Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
log("rawQueryParameters", queryParams);
// Console output: 'rawQueryParameters: {versionId=[abc123], partNumber=[77,
// 88]}'.

// Retrieve the first or all values for a query parameter as shown in the
// following code.
String versionId =
s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
log("firstMatchingRawQueryParameter-versionId", versionId);
// Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

String partNumber =
s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
log("firstMatchingRawQueryParameter-partNumber", partNumber);
// Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

List<String> partNumbers =
s3Uri.firstMatchingRawQueryParameters("partNumber");
log("firstMatchingRawQueryParameter", partNumbers);
// Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

/*
 * Object keys and query parameters with reserved or unsafe characters, must
be
```

```

    * URL-encoded.
    * For example replace whitespace " " with "%20".
    * Valid:
    * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
%5Bbrackets%5D"
    * Invalid:
    * "https://s3.us-west-1.amazonaws.com/myBucket/object key?query=[brackets]"
    *
    * Virtual-hosted-style URIs with bucket names that contain a dot, ".", the
dot
    * must not be URL-encoded.
    * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
    * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
    */
}

private static void log(String s3UriElement, Object element) {
    if (element == null) {
        logger.info("{}: {}", s3UriElement, "null");
    } else {
        logger.info("{}: {}", s3UriElement, element.toString());
    }
}
}

```

## Melakukan pengunggahan multibagian

Contoh kode berikut menunjukkan cara melakukan pengunggahan multibagian ke objek Amazon S3.

### SDK for Java 2.x

#### Note

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

Contoh kode menggunakan impor berikut.

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;

```

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

Gunakan [Manajer Transfer S3](#) di atas [klien S3 AWS berbasis CRT](#) untuk melakukan unggahan multibagian secara transparan ketika ukuran konten melebihi ambang batas. Ukuran ambang default adalah 8 MB.

```
public void multipartUploadWithTransferManager(String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key))
        .source(Paths.get(filePath))
        .build();
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
    fileUpload.completionFuture().join();
    transferManager.close();
}
```

Gunakan [S3Client API](#) atau (S3 AsyncClient API) untuk melakukan upload multipart.

```
public void multipartUploadWithS3Client(String filePath) {

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key));
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        int position = 0;
        while (position < fileSize) {
            file.seek(position);
            int read = file.getChannel().read(bb);

            bb.flip(); // Swap position and limit before reading from the
buffer.

            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .partNumber(partNumber)
                .build();

            UploadPartResponse partResponse = s3Client.uploadPart(
                uploadPartRequest,
                RequestBody.fromByteBuffer(bb));

            CompletedPart part = CompletedPart.builder()
                .partNumber(partNumber)
                .eTag(partResponse.eTag())
                .build();
            completedParts.add(part);

            bb.clear();
            position += read;
            partNumber++;
        }
    }
}
```

```
    }  
  } catch (IOException e) {  
    logger.error(e.getMessage());  
  }  
  
  // Complete the multipart upload.  
  s3Client.completeMultipartUpload(b -> b  
    .bucket(bucketName)  
    .key(key)  
    .uploadId(uploadId)  
  
    .multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));  
}
```


- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [CompleteMultipartUpload](#)
  - [CreateMultipartUpload](#)
  - [UploadPart](#)

Mengunggah atau mengunduh file besar

Contoh kode berikut menunjukkan cara mengunggah atau mengunduh file besar ke dan dari Amazon S3.

Untuk informasi selengkapnya, lihat [Pengunggahan objek menggunakan unggahan multibagian](#).

SDK for Java 2.x

 Note

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

Fungsi panggilan yang mentransfer file ke dan dari bucket S3 menggunakan TransferManager S3.

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,  
    URI destinationPathURI, String bucketName) {  
    DirectoryDownload directoryDownload =  
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
```

```

        .destination(Paths.get(destinationPathURI))
        .bucket(bucketName)
        .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}

```

Unggah seluruh direktori lokal.

```

public Integer uploadDirectory(S3TransferManager transferManager,
    URI sourceDirectory, String bucketName) {
    DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
        .source(Paths.get(sourceDirectory))
        .bucket(bucketName)
        .build());

    CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
    completedDirectoryUpload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryUpload.failedTransfers().size();
}

```

Unggah file tunggal.

```

public String uploadFile(S3TransferManager transferManager, String bucketName,
    String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create())
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
}

```

```
CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
return uploadResult.response().eTag();
}
```

## Mengunggah aliran ukuran yang tidak diketahui

Contoh kode berikut menunjukkan cara mengunggah aliran dengan ukuran yang tidak diketahui ke objek Amazon S3.

SDK for Java 2.x

### Note

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

Gunakan [Klien S3 berbasis CRT AWS](#).

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

/**
 * @param s3CrtAsyncClient - To upload content from a stream of unknown size,
 * use the AWS CRT-based S3 client. For more information, see
 * https://docs.aws.amazon.com/sdk-for-java/latest/
 * developer-guide/crt-based-s3-client.html.
 * @param bucketName - The name of the bucket.
 * @param key - The name of the object.
```

```
* @return software.amazon.awssdk.services.s3.model.PutObjectResponse - Returns
metadata pertaining to the put object operation.
*/
public PutObjectResponse putObjectFromStream(S3AsyncClient s3Client,
String bucketName, String key) {

    BlockingInputStreamAsyncRequestBody body =
        AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
stream will be provided later.

    CompletableFuture<PutObjectResponse> responseFuture =
        s3Client.putObject(r -> r.bucket(bucketName).key(key),
body);

    // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
randomString.length());

    // Provide the stream of data to be uploaded.
    body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

    PutObjectResponse response = responseFuture.join(); // Wait for the
response.
    logger.info("Object {} uploaded to bucket {}.", key, bucketName);
    return response;
}
}
```

### Gunakan [Manajer Transfer Amazon S3](#).

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;
```



```
import java.io.ByteArrayInputStream;
import java.util.UUID;

/**
 * @param transferManager - To upload content from a stream of unknown size, use
 the S3TransferManager based on the AWS CRT-based S3 client.
 *
 * For more information, see https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/transfer-manager.html.
 * @param bucketName - The name of the bucket.
 * @param key - The name of the object.
 * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
 result of the completed upload.
 */
public CompletedUpload uploadStream(S3TransferManager transferManager, String
bucketName, String key) {

    BlockingInputStreamAsyncRequestBody body =
        AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
stream will be provided later.

    Upload upload = transferManager.upload(builder -> builder
        .requestBody(body)
        .putObjectRequest(req -> req.bucket(bucketName).key(key))
        .build());

    // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
randomString.length());

    // Provide the stream of data to be uploaded.
    body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

    return upload.completionFuture().join();
}
}
```

## Gunakan checksum

Contoh kode berikut menunjukkan cara menggunakan checksum untuk bekerja dengan objek Amazon S3.

### SDK for Java 2.x

#### Note

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

Contoh kode menggunakan subset dari impor berikut.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
```

```
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

Tentukan algoritma checksum untuk metode `putObject` saat Anda [membangun `PutObjectRequest`](#).

```
public void putObjectWithChecksum() {
    s3Client.putObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(ChecksumAlgorithm.CRC32),
        RequestBody.fromString("This is a test"));
}
```

Verifikasi checksum untuk `getObject` metode saat Anda [membangun `GetObjectRequest`](#)

```
public GetObjectResponse getObjectWithChecksum() {
    return s3Client.getObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
        .response();
}
```

Hitung terlebih dahulu checksum untuk metode `putObject` saat Anda [membangun `PutObjectRequest`](#).

```
public void putObjectWithPrecalculatedChecksum(String filePath) {
    String checksum = calculateChecksum(filePath, "SHA-256");

    s3Client.putObject((b -> b
        .bucket(bucketName)
        .key(key)
        .checksumSHA256(checksum)),
        RequestBody.fromFile(Paths.get(filePath)));
}
```

```
}
```

Gunakan [Manajer Transfer S3](#) di atas [klien S3 AWS berbasis CRT](#) untuk melakukan unggahan multibagian secara transparan ketika ukuran konten melebihi ambang batas. Ukuran ambang default adalah 8 MB.

Anda dapat menentukan algoritma checksum untuk SDK yang akan digunakan. Secara default, SDK menggunakan algoritma CRC32.

```
public void multipartUploadWithChecksumTm(String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key)
            .checksumAlgorithm(ChecksumAlgorithm.SHA1))
        .source(Paths.get(filePath))
        .build();
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
    fileUpload.completionFuture().join();
    transferManager.close();
}
```

Gunakan [S3Client API](#) atau (S3 AsyncClient API) untuk melakukan upload multipart. Jika Anda menentukan checksum tambahan, Anda juga harus menentukan algoritma yang akan digunakan pada inisiasi unggahan. Anda juga harus menentukan algoritma untuk setiap permintaan bagian dan memberikan checksum yang dihitung untuk setiap bagian setelah diunggah.

```
public void multipartUploadWithChecksumS3Client(String filePath) {
    ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(algorithm)); // Checksum specified on initiation.
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
```

```

int partNumber = 1;
List<CompletedPart> completedParts = new ArrayList<>();
ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
    long fileSize = file.length();
    int position = 0;
    while (position < fileSize) {
        file.seek(position);
        int read = file.getChannel().read(bb);

        bb.flip(); // Swap position and limit before reading from the
buffer.

        UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .checksumAlgorithm(algorithm) // Checksum specified on each
part.

            .partNumber(partNumber)
            .build();

        UploadPartResponse partResponse = s3Client.uploadPart(
            uploadPartRequest,
            RequestBody.fromByteBuffer(bb));

        CompletedPart part = CompletedPart.builder()
            .partNumber(partNumber)
            .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.

            .eTag(partResponse.eTag())
            .build();
        completedParts.add(part);

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (IOException e) {
    System.err.println(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b

```

```
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)

        .multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
  - [CompleteMultipartUpload](#)
  - [CreateMultipartUpload](#)
  - [UploadPart](#)

## Contoh nirserver

Menginvokasi fungsi Lambda dari pemicu Amazon S3

Contoh kode berikut menunjukkan cara mengimplementasikan fungsi Lambda yang menerima peristiwa yang dipicu dengan mengunggah objek ke bucket S3. Fungsi ini mengambil nama bucket S3 dan kunci objek dari parameter peristiwa dan memanggil Amazon S3 API untuk mengambil dan mencatat jenis konten objek.

SDK for Java 2.x

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

Menggunakan peristiwa S3 dengan Lambda menggunakan Java.

```
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
```

```
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
    com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotificat

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
            String srcBucket = record.getS3().getBucket().getName();
            String srcKey = record.getS3().getObject().getUrlDecodedKey();

            S3Client s3Client = S3Client.builder().build();
            HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucket)
            .key(key)
            .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

## Contoh S3 Glacier menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan gletser AWS SDK for Java 2.x with S3.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

### Tindakan

Buat lemari besi

Contoh kode berikut menunjukkan cara membuat lemari besi Amazon S3 Glacier.

SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.CreateVaultRequest;
import software.amazon.awssdk.services.glacier.model.CreateVaultResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```



```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateVault {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <vaultName>

            Where:
                vaultName - The name of the vault to create.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void createGlacierVault(GlacierClient glacier, String vaultName) {
        try {
            CreateVaultRequest vaultRequest = CreateVaultRequest.builder()
                .vaultName(vaultName)
                .build();

            CreateVaultResponse createVaultResult =
glacier.createVault(vaultRequest);
            System.out.println("The URI of the new vault is " +
createVaultResult.location());

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
    }  
  }  
}
```

- Untuk detail API, lihat [CreateVault](#) di Referensi AWS SDK for Java 2.x API.

## Hapus brankas

Contoh kode berikut menunjukkan cara menghapus brankas Amazon S3 Glacier.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.glacier.GlacierClient;  
import software.amazon.awssdk.services.glacier.model.DeleteVaultRequest;  
import software.amazon.awssdk.services.glacier.model.GlacierException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class DeleteVault {  
    public static void main(String[] args) {  
  
        final String usage = ""  
  
            Usage:    <vaultName>  
  
            Where:  
                vaultName - The name of the vault to delete.\s
```

```
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String vaultName = args[0];
    GlacierClient glacier = GlacierClient.builder()
        .region(Region.US_EAST_1)
        .build();

    deleteGlacierVault(glacier, vaultName);
    glacier.close();
}

public static void deleteGlacierVault(GlacierClient glacier, String vaultName) {
    try {
        DeleteVaultRequest delVaultRequest = DeleteVaultRequest.builder()
            .vaultName(vaultName)
            .build();

        glacier.deleteVault(delVaultRequest);
        System.out.println("The vault was deleted!");


    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DeleteVault](#) di Referensi AWS SDK for Java 2.x API.

## Hapus arsip

Contoh kode berikut menunjukkan cara menghapus arsip Amazon S3 Glacier.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteArchiveRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteArchive {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <vaultName> <accountId> <archiveId>

            Where:
                vaultName - The name of the vault that contains the archive to
delete.

                accountId - The account ID value.
                archiveId - The archive ID value.

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        String accountId = args[1];
        String archiveId = args[2];
```

```
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();

        deleteGlacierArchive(glacier, vaultName, accountId, archiveId);
        glacier.close();
    }

    public static void deleteGlacierArchive(GlacierClient glacier, String vaultName,
String accountId,
        String archiveId) {
        try {
            DeleteArchiveRequest delArcRequest = DeleteArchiveRequest.builder()
                .vaultName(vaultName)
                .accountId(accountId)
                .archiveId(archiveId)
                .build();

            glacier.deleteArchive(delArcRequest);
            System.out.println("The archive was deleted.");

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [DeleteArchive](#) di Referensi AWS SDK for Java 2.x API.

## Daftar brankas

Contoh kode berikut menunjukkan cara membuat daftar brankas Amazon S3 Glacier.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.model.ListVaultsRequest;
import software.amazon.awssdk.services.glacier.model.ListVaultsResponse;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DescribeVaultOutput;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListVaults {
    public static void main(String[] args) {
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllVault(glacier);
        glacier.close();
    }

    public static void listAllVault(GlacierClient glacier) {
        boolean listComplete = false;
        String newMarker = null;
        int totalVaults = 0;
        System.out.println("Your Amazon Glacier vaults:");
        try {
            while (!listComplete) {
                ListVaultsResponse response = null;
                if (newMarker != null) {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                        .marker(newMarker)
                        .build();

                    response = glacier.listVaults(request);
                } else {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                        .build();
                }
            }
        }
    }
}
```

```
        response = glacier.listVaults(request);
    }

    List<DescribeVaultOutput> vaultList = response.vaultList();
    for (DescribeVaultOutput v : vaultList) {
        totalVaults += 1;
        System.out.println("* " + v.vaultName());
    }

    // Check for further results.
    newMarker = response.marker();
    if (newMarker == null) {
        listComplete = true;
    }
}

if (totalVaults == 0) {
    System.out.println("No vaults found.");
}

} catch (GlacierException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
}
```

- Untuk detail API, lihat [ListVaults](#) di Referensi AWS SDK for Java 2.x API.

## Mengambil inventaris vault

Contoh kode berikut menunjukkan cara mengambil inventaris Amazon S3 Glacier vault.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.JobParameters;
import software.amazon.awssdk.services.glacier.model.InitiateJobResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import software.amazon.awssdk.services.glacier.model.InitiateJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobResponse;
import software.amazon.awssdk.services.glacier.model.GetJobOutputRequest;
import software.amazon.awssdk.services.glacier.model.GetJobOutputResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ArchiveDownload {
    public static void main(String[] args) {

        final String usage = ""

            Usage:    <vaultName> <accountId> <path>

            Where:
                vaultName - The name of the vault.
                accountId - The account ID value.
                path - The path where the file is written to.
            "";

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
```



```
String accountId = args[1];
String path = args[2];
GlacierClient glacier = GlacierClient.builder()
    .region(Region.US_EAST_1)
    .build();

String jobNum = createJob(glacier, vaultName, accountId);
checkJob(glacier, jobNum, vaultName, accountId, path);
glacier.close();
}

public static String createJob(GlacierClient glacier, String vaultName, String
accountId) {
    try {
        JobParameters job = JobParameters.builder()
            .type("inventory-retrieval")
            .build();

        InitiateJobRequest initJob = InitiateJobRequest.builder()
            .jobParameters(job)
            .accountId(accountId)
            .vaultName(vaultName)
            .build();

        InitiateJobResponse response = glacier.initiateJob(initJob);
        System.out.println("The job ID is: " + response.jobId());
        System.out.println("The relative URI path of the job is: " +
response.location());
        return response.jobId();

    } catch (GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

// Poll S3 Glacier = Polling a Job may take 4-6 hours according to the
// Documentation.
public static void checkJob(GlacierClient glacier, String jobId, String name,
String account, String path) {
    try {
        boolean finished = false;
```

```
String jobStatus;
int yy = 0;

while (!finished) {
    DescribeJobRequest jobRequest = DescribeJobRequest.builder()
        .jobId(jobId)
        .accountId(account)
        .vaultName(name)
        .build();

    DescribeJobResponse response = glacier.describeJob(jobRequest);
    jobStatus = response.statusCodeAsString();

    if (jobStatus.compareTo("Succeeded") == 0)
        finished = true;
    else {
        System.out.println(yy + " status is: " + jobStatus);
        Thread.sleep(1000);
    }
    yy++;
}

System.out.println("Job has Succeeded");
GetJobOutputRequest jobOutputRequest = GetJobOutputRequest.builder()
    .jobId(jobId)
    .vaultName(name)
    .accountId(account)
    .build();

ResponseBytes<GetJobOutputResponse> objectBytes =
glacier.getJobOutputAsBytes(jobOutputRequest);
// Write the data to a local file.
byte[] data = objectBytes.asByteArray();
File myFile = new File(path);
OutputStream os = new FileOutputStream(myFile);
os.write(data);
System.out.println("Successfully obtained bytes from a Glacier vault");
os.close();

} catch (GlacierException | InterruptedException | IOException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```


```
}  
}
```

- Untuk detail API, lihat [InitiateJob](#) di Referensi AWS SDK for Java 2.x API.

Unggah arsip ke brankas

Contoh kode berikut menunjukkan cara mengunggah arsip ke brankas Amazon S3 Glacier.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.glacier.GlacierClient;  
import software.amazon.awssdk.services.glacier.model.UploadArchiveRequest;  
import software.amazon.awssdk.services.glacier.model.UploadArchiveResponse;  
import software.amazon.awssdk.services.glacier.model.GlacierException;  
import java.io.File;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.security.MessageDigest;  
import java.security.NoSuchAlgorithmException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class UploadArchive {  
  
    static final int ONE_MB = 1024 * 1024;
```

```
public static void main(String[] args) {
    final String usage = ""

        Usage:  <strPath> <vaultName>\s

        Where:
            strPath - The path to the archive to upload (for example, C:\\AWS
\\test.pdf).
            vaultName - The name of the vault.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String strPath = args[0];
    String vaultName = args[1];
    File myFile = new File(strPath);
    Path path = Paths.get(strPath);
    GlacierClient glacier = GlacierClient.builder()
        .region(Region.US_EAST_1)
        .build();

    String archiveId = uploadContent(glacier, path, vaultName, myFile);
    System.out.println("The ID of the archived item is " + archiveId);
    glacier.close();
}

public static String uploadContent(GlacierClient glacier, Path path, String
vaultName, File myFile) {
    // Get an SHA-256 tree hash value.
    String checkVal = computeSHA256(myFile);
    try {
        UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
            .vaultName(vaultName)
            .checksum(checkVal)
            .build();

        UploadArchiveResponse res = glacier.uploadArchive(uploadRequest, path);
        return res.archiveId();
    } catch (GlacierException e) {
```

```

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

private static String computeSHA256(File inputFile) {
    try {
        byte[] treeHash = computeSHA256TreeHash(inputFile);
        System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
        return toHex(treeHash);

    } catch (IOException ioe) {
        System.err.format("Exception when reading from file %s: %s", inputFile,
ioe.getMessage());
        System.exit(-1);

    } catch (NoSuchAlgorithmException nsae) {
        System.err.format("Cannot locate MessageDigest algorithm for SHA-256:
%s", nsae.getMessage());
        System.exit(-1);
    }
    return "";
}

public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
    NoSuchAlgorithmException {

    byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
    return computeSHA256TreeHash(chunkSHA256Hashes);
}

/**
 * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
 * includes the checksum for the last chunk, even if it's smaller than 1 MB.
 */
public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
    NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    long numChunks = file.length() / ONE_MB;
    if (file.length() % ONE_MB > 0) {
        numChunks++;
    }
}

```

```

    if (numChunks == 0) {
        return new byte[][] { md.digest() };
    }

    byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
    FileInputStream fileStream = null;

    try {
        fileStream = new FileInputStream(file);
        byte[] buff = new byte[ONE_MB];

        int bytesRead;
        int idx = 0;

        while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
            md.reset();
            md.update(buff, 0, bytesRead);
            chunkSHA256Hashes[idx++] = md.digest();
        }

        return chunkSHA256Hashes;
    } finally {
        if (fileStream != null) {
            try {
                fileStream.close();
            } catch (IOException ioe) {
                System.err.printf("Exception while closing %s.\n %s",
file.getName(),
                                ioe.getMessage());
            }
        }
    }
}

/**
 * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
 * checksums.
 */
public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
    throws NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");

```

```

byte[][] prevLvlHashes = chunkSHA256Hashes;
while (prevLvlHashes.length > 1) {
    int len = prevLvlHashes.length / 2;
    if (prevLvlHashes.length % 2 != 0) {
        len++;
    }

    byte[][] currLvlHashes = new byte[len][];
    int j = 0;
    for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

        // If there are at least two elements remaining.
        if (prevLvlHashes.length - i > 1) {

            // Calculate a digest of the concatenated nodes.
            md.reset();
            md.update(prevLvlHashes[i]);
            md.update(prevLvlHashes[i + 1]);
            currLvlHashes[j] = md.digest();

        } else { // Take care of the remaining odd chunk
            currLvlHashes[j] = prevLvlHashes[i];
        }
    }

    prevLvlHashes = currLvlHashes;
}

return prevLvlHashes[0];
}

/**
 * Returns the hexadecimal representation of the input byte array
 */
public static String toHex(byte[] data) {
    StringBuilder sb = new StringBuilder(data.length * 2);
    for (byte datum : data) {
        String hex = Integer.toHexString(datum & 0xFF);

        if (hex.length() == 1) {
            // Append leading zero.
            sb.append("0");
        }
        sb.append(hex);
    }
}

```

```
    }  
    return sb.toString().toLowerCase();  
  }  
}
```

- Untuk detail API, lihat [UploadArchive](#) di Referensi AWS SDK for Java 2.x API.

## SageMaker contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with SageMaker.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Memulai

Halo SageMaker

Contoh kode berikut menunjukkan cara untuk mulai menggunakan SageMaker.

SDK for Java 2.x

### Note

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

```
/**  
 * Before running this Java V2 code example, set up your development
```



```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class HelloSageMaker {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SageMakerClient sageMakerClient = SageMakerClient.builder()
            .region(region)
            .build();

        listBooks(sageMakerClient);
        sageMakerClient.close();
    }

    public static void listBooks(SageMakerClient sageMakerClient) {
        try {
            ListNotebookInstancesResponse notebookInstancesResponse =
sageMakerClient.listNotebookInstances();
            List<NotebookInstanceSummary> items =
notebookInstancesResponse.notebookInstances();
            for (NotebookInstanceSummary item : items) {
                System.out.println("The notebook name is: " +
item.notebookInstanceName());
            }

        } catch (SageMakerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListNotebookInstances](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

### Buat pipeline

Contoh kode berikut menunjukkan cara membuat atau memperbarui pipeline di SageMaker.

### SDK for Java 2.x

#### Note

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

```
// Create a pipeline from the example pipeline JSON.
public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
    String functionArn, String pipelineName) {
    System.out.println("Setting up the pipeline.");
    JSONParser parser = new JSONParser();

    // Read JSON and get pipeline definition.
    try (FileReader reader = new FileReader(filePath)) {
        Object obj = parser.parse(reader);
        JSONObject jsonObject = (JSONObject) obj;
        JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
        for (Object stepObj : stepsArray) {
            JSONObject step = (JSONObject) stepObj;
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArn);
            }
        }
        System.out.println(jsonObject);

        // Create the pipeline.
        CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
            .pipelineDescription("Java SDK example pipeline")
            .roleArn(roleArn)
            .pipelineName(pipelineName)
            .pipelineDefinition(jsonObject.toString())
            .build();
    }
}
```

```
sageMakerClient.createPipeline(pipelineRequest);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
} catch (IOException | ParseException e) {
    throw new RuntimeException(e);
}
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [CreatePipeline](#)
  - [UpdatePipeline](#)

## Hapus pipa

Contoh kode berikut menunjukkan cara menghapus pipeline di SageMaker.

### SDK for Java 2.x

#### Note

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

```
// Delete a SageMaker pipeline by name.
public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
    DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
        .pipelineName(pipelineName)
        .build();

    sageMakerClient.deletePipeline(pipelineRequest);
    System.out.println("*** Successfully deleted " + pipelineName);
}
```

- Untuk detail API, lihat [DeletePipeline](#) di Referensi AWS SDK for Java 2.x API.

## Jelaskan eksekusi pipeline

Contoh kode berikut menunjukkan bagaimana mendeskripsikan eksekusi pipeline di SageMaker.

SDK for Java 2.x

### Note

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

```
// Check the status of a pipeline execution.
public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
    throws InterruptedException {
    String status;
    int index = 0;
    do {
        DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
            .pipelineExecutionArn(executionArn)
            .build();

        DescribePipelineExecutionResponse response = sageMakerClient
            .describePipelineExecution(pipelineExecutionRequest);
        status = response.pipelineExecutionStatusAsString();
        System.out.println(index + ". The Status of the pipeline is " + status);
        TimeUnit.SECONDS.sleep(4);
        index++;
    } while ("Executing".equals(status));
    System.out.println("Pipeline finished with status " + status);
}
```

- Untuk detail API, lihat [DescribePipelineExecution](#) di Referensi AWS SDK for Java 2.x API.

## Jalankan pipa

Contoh kode berikut menunjukkan cara memulai eksekusi pipeline di SageMaker.

## SDK for Java 2.x

 Note

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

```
// Start a pipeline run with job configurations.
public static String executePipeline(SageMakerClient sageMakerClient, String
bucketName, String queueUrl,
    String roleArn, String pipelineName) {
    System.out.println("Starting pipeline execution.");
    String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
    String output = "s3://" + bucketName + "/outputfiles/";
    Gson gson = new GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting().create();

    // Set up all parameters required to start the pipeline.
    List<Parameter> parameters = new ArrayList<>();
    Parameter para1 = Parameter.builder()
        .name("parameter_execution_role")
        .value(roleArn)
        .build();

    Parameter para2 = Parameter.builder()
        .name("parameter_queue_url")
        .value(queueUrl)
        .build();

    String inputJSON = "{\n" +
        "  \"DataSourceConfig\": {\n" +
        "    \"S3Data\": {\n" +
        "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
        "    },\n" +
        "    \"Type\": \"S3_DATA\"\n" +
        "  },\n" +
        "  \"DocumentType\": \"CSV\"\n" +
        "}";
```

```
System.out.println(inputJSON);

Parameter para3 = Parameter.builder()
    .name("parameter_vej_input_config")
    .value(inputJSON)
    .build();

// Create an ExportVectorEnrichmentJobOutputConfig object.
VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
    .s3Uri(output)
    .build();

ExportVectorEnrichmentJobOutputConfig outputConfig =
ExportVectorEnrichmentJobOutputConfig.builder()
    .s3Data(jobS3Data)
    .build();

String gson4 = gson.toJson(outputConfig);
Parameter para4 = Parameter.builder()
    .name("parameter_vej_export_config")
    .value(gson4)
    .build();

System.out.println("parameter_vej_export_config:" +
gson.toJson(outputConfig));

// Create a VectorEnrichmentJobConfig object.
ReverseGeocodingConfig reverseGeocodingConfig =
ReverseGeocodingConfig.builder()
    .xAttributeName("Longitude")
    .yAttributeName("Latitude")
    .build();

VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
    .reverseGeocodingConfig(reverseGeocodingConfig)
    .build();

String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
Parameter para5 = Parameter.builder()
    .name("parameter_step_1_vej_config")
    .value(para5JSON)
    .build();
```

```
System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
parameters.add(para1);
parameters.add(para2);
parameters.add(para3);
parameters.add(para4);
parameters.add(para5);

StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
    .pipelineExecutionDescription("Created using Java SDK")
    .pipelineExecutionDisplayName(pipelineName + "-example-execution")
    .pipelineParameters(parameters)
    .pipelineName(pipelineName)
    .build();

StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
return response.pipelineExecutionArn();
}
```

- Untuk detail API, lihat [StartPipelineExecution](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

Memulai pekerjaan geospasial dan jaringan pipa

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- 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 SDK di Community.aws](#).

## SDK for Java 2.x

 Note

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

```
public class SagemakerWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String eventSourceMapping = "";

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
            "Usage:\n" +
            "    <sageMakerRoleName> <lambdaRoleName> <functionFileLocation>
<functionName> <queueName> <bucketName> <lnglatData> <spatialPipelinePath>
<pipelineName>\n\n"
            +
            "Where:\n" +
            "    sageMakerRoleName - The name of the Amazon SageMaker role.\n\n"
+
            "    lambdaRoleName - The name of the AWS Lambda role.\n\n" +
            "    functionFileLocation - The file location where the JAR file
that represents the AWS Lambda function is located.\n\n"
            +
            "    functionName - The name of the AWS Lambda function (for
example, SageMakerExampleFunction).\n\n" +
            "    queueName - The name of the Amazon Simple Queue Service (Amazon
SQS) queue.\n\n" +
            "    bucketName - The name of the Amazon Simple Storage Service
(Amazon S3) bucket.\n\n" +
            "    lnglatData - The file location of the latlongtest.csv file
required for this use case.\n\n" +
            "    spatialPipelinePath - The file location of the
GeoSpatialPipeline.json file required for this use case.\n\n"
            +
            "    pipelineName - The name of the pipeline to create (for example,
sagemaker-sdk-example-pipeline).\n\n";

        if (args.length != 9) {
            System.out.println(usage);
        }
    }
}
```



```
        System.exit(1);
    }

    String sageMakerRoleName = args[0];
    String lambdaRoleName = args[1];
    String functionFileLocation = args[2];
    String functionName = args[3];
    String queueName = args[4];
    String bucketName = args[5];
    String lnglatData = args[6];
    String spatialPipelinePath = args[7];
    String pipelineName = args[8];
    String handlerName = "org.example.SageMakerLambdaFunction::handleRequest";

    Region region = Region.US_WEST_2;
    SageMakerClient sageMakerClient = SageMakerClient.builder()
        .region(region)
        .build();

    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    LambdaClient lambdaClient = LambdaClient.builder()
        .region(region)
        .build();

    SqsClient sqsClient = SqsClient.builder()
        .region(region)
        .build();

    S3Client s3Client = S3Client.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon SageMaker pipeline example
scenario.");
    System.out.println(
        "\nThis example workflow will guide you through setting up and
running an" +
            "\nAmazon SageMaker pipeline. The pipeline uses an AWS
Lambda function and an" +
```

```
        "\nAmazon SQS Queue. It runs a vector enrichment reverse
geocode job to" +
        "\nreverse geocode addresses in an input file and store the
results in an export file.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("First, we will set up the roles, functions, and queue
needed by the SageMaker pipeline.");
    String lambdaRoleArn = checkLambdaRole(iam, lambdaRoleName);
    String sageMakerRoleArn = checkSageMakerRole(iam, sageMakerRoleName);

    String functionArn = checkFunction(lambdaClient, functionName,
functionFileLocation, lambdaRoleArn,
        handlerName);
    String queueUrl = checkQueue(sqsClient, lambdaClient, queueName,
functionName);
    System.out.println("The queue URL is " + queueUrl);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Setting up bucket " + bucketName);
    if (!checkBucket(s3Client, bucketName)) {
        setupBucket(s3Client, bucketName);
        System.out.println("Put " + lnglatData + " into " + bucketName);
        putS3Object(s3Client, bucketName, "latlongtest.csv", lnglatData);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Now we can create and run our pipeline.");
    setupPipeline(sageMakerClient, spatialPipelinePath, sageMakerRoleArn,
functionArn, pipelineName);
    String pipelineExecutionARN = executePipeline(sageMakerClient, bucketName,
queueUrl, sageMakerRoleArn,
        pipelineName);
    System.out.println("The pipeline execution ARN value is " +
pipelineExecutionARN);
    waitForPipelineExecution(sageMakerClient, pipelineExecutionARN);
    System.out.println("Getting output results " + bucketName);
    getOutputResults(s3Client, bucketName);
    System.out.println(DASHES);

    System.out.println(DASHES);
```

```

        System.out.println("The pipeline has completed. To view the pipeline and
runs " +
        "in SageMaker Studio, follow these instructions:" +
        "\nhttps://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-
studio.html");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Do you want to delete the AWS resources used in this
Workflow? (y/n)");
        Scanner in = new Scanner(System.in);
        String delResources = in.nextLine();
        if (delResources.compareTo("y") == 0) {
            System.out.println("Lets clean up the AWS resources. Wait 30 seconds");
            TimeUnit.SECONDS.sleep(30);
            deleteEventSourceMapping(lambdaClient);
            deleteSQSQueue(sqsClient, queueName);
            listBucketObjects(s3Client, bucketName);
            deleteBucket(s3Client, bucketName);
            deleteLambdaFunction(lambdaClient, functionName);
            deleteLambdaRole(iam, lambdaRoleName);
            deleteSagemakerRole(iam, sageMakerRoleName);
            deletePipeline(sageMakerClient, pipelineName);
        } else {
            System.out.println("The AWS Resources were not deleted!");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SageMaker pipeline scenario is complete.");
        System.out.println(DASHES);
    }

    private static void readObject(S3Client s3Client, String bucketName, String key)
    {
        System.out.println("Output file contents: \n");
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
        byte[] byteArray = objectBytes.asByteArray();
    }

```

```
String text = new String(byteArray, StandardCharsets.UTF_8);
System.out.println("Text output: " + text);
}

// Display some results from the output directory.
public static void getOutputResults(S3Client s3Client, String bucketName) {
    System.out.println("Getting output results {bucketName}.");
    ListObjectsRequest listObjectsRequest = ListObjectsRequest.builder()
        .bucket(bucketName)
        .prefix("outputfiles/")
        .build();

    ListObjectsResponse response = s3Client.listObjects(listObjectsRequest);
    List<S3Object> s3Objects = response.contents();
    for (S3Object object : s3Objects) {
        readObject(s3Client, bucketName, object.key());
    }
}

// Check the status of a pipeline execution.
public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
    throws InterruptedException {
    String status;
    int index = 0;
    do {
        DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
            .pipelineExecutionArn(executionArn)
            .build();

        DescribePipelineExecutionResponse response = sageMakerClient
            .describePipelineExecution(pipelineExecutionRequest);
        status = response.pipelineExecutionStatusAsString();
        System.out.println(index + ". The Status of the pipeline is " + status);
        TimeUnit.SECONDS.sleep(4);
        index++;
    } while ("Executing".equals(status));
    System.out.println("Pipeline finished with status " + status);
}

// Delete a SageMaker pipeline by name.
public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
```

```
DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
    .pipelineName(pipelineName)
    .build();

sageMakerClient.deletePipeline(pipelineRequest);
System.out.println("*** Successfully deleted " + pipelineName);
}

// Create a pipeline from the example pipeline JSON.
public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
    String functionArn, String pipelineName) {
    System.out.println("Setting up the pipeline.");
    JSONParser parser = new JSONParser();

    // Read JSON and get pipeline definition.
    try (FileReader reader = new FileReader(filePath)) {
        Object obj = parser.parse(reader);
        JSONObject jsonObject = (JSONObject) obj;
        JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
        for (Object stepObj : stepsArray) {
            JSONObject step = (JSONObject) stepObj;
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArn);
            }
        }
    }
    System.out.println(jsonObject);

    // Create the pipeline.
    CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
        .pipelineDescription("Java SDK example pipeline")
        .roleArn(roleArn)
        .pipelineName(pipelineName)
        .pipelineDefinition(jsonObject.toString())
        .build();

    sageMakerClient.createPipeline(pipelineRequest);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
} catch (IOException | ParseException e) {
    throw new RuntimeException(e);
}
```

```
}

// Start a pipeline run with job configurations.
public static String executePipeline(SageMakerClient sageMakerClient, String
bucketName, String queueUrl,
    String roleArn, String pipelineName) {
    System.out.println("Starting pipeline execution.");
    String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
    String output = "s3://" + bucketName + "/outputfiles/";
    Gson gson = new GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting().create();

    // Set up all parameters required to start the pipeline.
    List<Parameter> parameters = new ArrayList<>();
    Parameter para1 = Parameter.builder()
        .name("parameter_execution_role")
        .value(roleArn)
        .build();

    Parameter para2 = Parameter.builder()
        .name("parameter_queue_url")
        .value(queueUrl)
        .build();

    String inputJSON = "{\n" +
        "  \"DataSourceConfig\": {\n" +
        "    \"S3Data\": {\n" +
        "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
        "    },\n" +
        "    \"Type\": \"S3_DATA\"\n" +
        "  },\n" +
        "  \"DocumentType\": \"CSV\"\n" +
        "}";

    System.out.println(inputJSON);

    Parameter para3 = Parameter.builder()
        .name("parameter_vej_input_config")
        .value(inputJSON)
        .build();
```

```
// Create an ExportVectorEnrichmentJobOutputConfig object.
VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
    .s3Uri(output)
    .build();

ExportVectorEnrichmentJobOutputConfig outputConfig =
ExportVectorEnrichmentJobOutputConfig.builder()
    .s3Data(jobS3Data)
    .build();

String gson4 = gson.toJson(outputConfig);
Parameter para4 = Parameter.builder()
    .name("parameter_vej_export_config")
    .value(gson4)
    .build();

System.out.println("parameter_vej_export_config:" +
gson.toJson(outputConfig));

// Create a VectorEnrichmentJobConfig object.
ReverseGeocodingConfig reverseGeocodingConfig =
ReverseGeocodingConfig.builder()
    .xAttributeName("Longitude")
    .yAttributeName("Latitude")
    .build();

VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
    .reverseGeocodingConfig(reverseGeocodingConfig)
    .build();

String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
Parameter para5 = Parameter.builder()
    .name("parameter_step_1_vej_config")
    .value(para5JSON)
    .build();

System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
parameters.add(para1);
parameters.add(para2);
parameters.add(para3);
parameters.add(para4);
parameters.add(para5);
```

```
        StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
    .pipelineExecutionDescription("Created using Java SDK")
    .pipelineExecutionDisplayName(pipelineName + "-example-execution")
    .pipelineParameters(parameters)
    .pipelineName(pipelineName)
    .build();

        StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
        return response.pipelineExecutionArn();
    }

    public static void deleteEventSourceMapping(LambdaClient lambdaClient) {
        DeleteEventSourceMappingRequest eventSourceMappingRequest =
DeleteEventSourceMappingRequest.builder()
    .uuid(eventSourceMapping)
    .build();

        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest);
    }

    public static void deleteSagemakerRole(IamClient iam, String roleName) {
        String[] sageMakerRolePolicies = getSageMakerRolePolicies();
        try {
            for (String policy : sageMakerRolePolicies) {
                // First the policy needs to be detached.
                DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
    .policyArn(policy)
    .roleName(roleName)
    .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
    .roleName(roleName)
    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);
        }
    }
}
```



```
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteLambdaRole(IamClient iam, String roleName) {
    String[] lambdaRolePolicies = getLambdaRolePolicies();
    try {
        for (String policy : lambdaRolePolicies) {
            // First the policy needs to be detached.
            DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                .policyArn(policy)
                .roleName(roleName)
                .build();

            iam.detachRolePolicy(rolePolicyRequest);
        }

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Delete the specific AWS Lambda function.
public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("*** " + functionName + " was deleted");
    }
}
```

```
    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Delete the specific S3 bucket.
public static void deleteBucket(S3Client s3Client, String bucketName) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucketName)
        .build();
    s3Client.deleteBucket(deleteBucketRequest);
    System.out.println("*** " + bucketName + " was deleted.");
}

public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.print("\n The name of the key is " + myValue.key());
            deleteBucketObjects(s3, bucketName, myValue.key());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteBucketObjects(S3Client s3, String bucketName, String
objectName) {
    ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
    toDelete.add(ObjectIdentifier.builder()
        .key(objectName)
        .build());
    try {
        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
```

```
        .bucket(bucketName)
        .delete(Delete.builder()
            .objects(toDelete).build())
        .build();

    s3.deleteObjects(dor);
    System.out.println("*** " + bucketName + " objects were deleted.");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

// Delete the specific Amazon SQS queue.
public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
    try {
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
    try {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("x-amz-meta-myVal", "test");
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key("samplefiles/" + objectKey)
            .metadata(metadata)
            .build();
```

```
s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void setupBucket(S3Client s3Client, String bucketName) {
    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Set up the SQS queue to use with the pipeline.
public static String setupQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
    String lambdaName) {
    System.out.println("Setting up queue named " + queueName);
    try {
        Map<QueueAttributeName, String> queueAtt = new HashMap<>();
        queueAtt.put(QueueAttributeName.DELAY_SECONDS, "5");
        queueAtt.put(QueueAttributeName.RECEIVE_MESSAGE_WAIT_TIME_SECONDS, "5");
```

```

        queueAtt.put(QueueAttributeName.VISIBILITY_TIMEOUT, "300");
        CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
            .queueName(queueName)
            .attributes(queueAtt)
            .build();

        sqsClient.createQueue(createQueueRequest);
        System.out.println("\nGet queue url");
        GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        TimeUnit.SECONDS.sleep(15);

        connectLambda(sqsClient, lambdaClient, getQueueUrlResponse.queueUrl(),
lambdaName);
        System.out.println("Queue ready with Url " +
getQueueUrlResponse.queueUrl());
        return getQueueUrlResponse.queueUrl();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    return "";
}

// Connect the queue to the Lambda function as an event source.
public static void connectLambda(SqsClient sqsClient, LambdaClient lambdaClient,
String queueUrl,
    String lambdaName) {
    System.out.println("Connecting the Lambda function and queue for the
pipeline.");
    String queueArn = "";

    // Specify the attributes to retrieve.
    List<QueueAttributeName> atts = new ArrayList<>();
    atts.add(QueueAttributeName.QUEUE_ARN);
    GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
        .queueUrl(queueUrl)
        .attributeNames(atts)
        .build();

```

```
    GetQueueAttributesResponse response =
sqscClient.getQueueAttributes(attributesRequest);
    Map<String, String> queueAtts = response.attributesAsStrings();
    for (Map.Entry<String, String> queueAtt : queueAtts.entrySet()) {
        System.out.println("Key = " + queueAtt.getKey() + ", Value = " +
queueAtt.getValue());
        queueArn = queueAtt.getValue();
    }

    CreateEventSourceMappingRequest eventSourceMappingRequest =
CreateEventSourceMappingRequest.builder()
        .eventSourceArn(queueArn)
        .functionName(lambdaName)
        .build();

    CreateEventSourceMappingResponse response1 =
lambdaClient.createEventSourceMapping(eventSourceMappingRequest);
    eventSourceMapping = response1.uuid();
    System.out.println("The mapping between the event source and Lambda function
was successful");
}

// Create an AWS Lambda function.
public static String createLambdaFunction(LambdaClient awsLambda, String
functionName, String filePath, String role,
String handler) {
    try {
        LambdaWaiter waiter = awsLambda.waiter();
        InputStream is = new FileInputStream(filePath);
        SdkBytes fileToUpload = SdkBytes.fromInputStream(is);
        FunctionCode code = FunctionCode.builder()
            .zipFile(fileToUpload)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .functionName(functionName)
            .description("SageMaker example function.")
            .code(code)
            .handler(handler)
            .runtime(Runtime.JAVA11)
            .timeout(200)
            .memorySize(1024)
            .role(role)
    }
```

```

        .build());

        // Create a Lambda function using a waiter.
        CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
        GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();
        WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The function ARN is " +
functionResponse.functionArn());
        return functionResponse.functionArn();

    } catch (LambdaException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static String createSageMakerRole(IamClient iam, String roleName) {
    String[] sageMakerRolePolicies = getSageMakerRolePolicies();
    System.out.println("Creating a role to use with SageMaker.");
    String assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\", " +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\", " +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\", " +
        "\"sagemaker-geospatial.amazonaws.com\", " +
        "\"lambda.amazonaws.com\", " +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}, " +
        "\"Action\": \"sts:AssumeRole\"" +
        "]}";

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(roleName)

```

```

        .assumeRolePolicyDocument(assumeRolePolicy)
        .description("Created using the AWS SDK for Java")
        .build();

    CreateRoleResponse roleResult = iam.createRole(request);

    // Attach the policies to the role.
    for (String policy : sageMakerRolePolicies) {
        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(policy)
        .build();

        iam.attachRolePolicy(attachRequest);
    }

    // Allow time for the role to be ready.
    TimeUnit.SECONDS.sleep(15);
    System.out.println("Role ready with ARN " + roleResult.role().arn());
    return roleResult.role().arn();

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
} catch (InterruptedException e) {
    throw new RuntimeException(e);
}
return "";
}

private static String createLambdaRole(IamClient iam, String roleName) {
    String[] lambdaRolePolicies = getLambdaRolePolicies();
    String assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\", " +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\", " +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\", " +
        "\"sagemaker-geospatial.amazonaws.com\", " +
        "\"lambda.amazonaws.com\", " +
        "\"s3.amazonaws.com\"" +
        "]" +
    "}" +

```



```

        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]"+
        "}";

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(roleName)
            .assumeRolePolicyDocument(assumeRolePolicy)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse roleResult = iam.createRole(request);

        // Attach the policies to the role.
        for (String policy : lambdaRolePolicies) {
            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policy)
                .build();

            iam.attachRolePolicy(attachRequest);
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15);
        System.out.println("Role ready with ARN " + roleResult.role().arn());
        return roleResult.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }

    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    }
    return "";
}

public static String checkFunction(LambdaClient lambdaClient, String
functionName, String filePath, String role,
String handler) {
    System.out.println("Create an AWS Lambda function used in this workflow.");
    String functionArn;

```

```
    try {
        // Does this function already exist.
        GetFunctionRequest functionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();

        GetFunctionResponse response =
lambdaClient.getFunction(functionRequest);
        functionArn = response.configuration().functionArn();

    } catch (LambdaException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        functionArn = createLambdaFunction(lambdaClient, functionName, filePath,
role, handler);
    }
    return functionArn;
}

// Check to see if the specific S3 bucket exists. If the S3 bucket exists, this
// method returns true.
public static boolean checkBucket(S3Client s3, String bucketName) {
    try {
        HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3.headBucket(headBucketRequest);
        System.out.println(bucketName + " exists");
        return true;

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    return false;
}

// Checks to see if the Amazon SQS queue exists. If not, this method creates a
// new queue
// and returns the ARN value.
public static String checkQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
    String lambdaName) {
    System.out.println("Creating a queue for this use case.");
    String queueUrl;
```

```
    try {
        GetQueueUrlRequest request = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        GetQueueUrlResponse response = sqsClient.getQueueUrl(request);
        queueUrl = response.queueUrl();
        System.out.println(queueUrl);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        queueUrl = setupQueue(sqsClient, lambdaClient, queueName, lambdaName);
    }
    return queueUrl;
}

// Checks to see if the Lambda role exists. If not, this method creates it.
public static String checkLambdaRole(IamClient iam, String roleName) {
    System.out.println("Creating a role to for AWS Lambda to use.");
    String roleArn;
    try {
        GetRoleRequest roleRequest = GetRoleRequest.builder()
            .roleName(roleName)
            .build();

        GetRoleResponse response = iam.getRole(roleRequest);
        roleArn = response.role().arn();
        System.out.println(roleArn);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        roleArn = createLambdaRole(iam, roleName);
    }
    return roleArn;
}

// Checks to see if the SageMaker role exists. If not, this method creates it.
public static String checkSageMakerRole(IamClient iam, String roleName) {
    System.out.println("Creating a role to for AWS SageMaker to use.");
    String roleArn;
    try {
        GetRoleRequest roleRequest = GetRoleRequest.builder()
            .roleName(roleName)
            .build();
```

```

        GetRoleResponse response = iam.getRole(roleRequest);
        roleArn = response.role().arn();
        System.out.println(roleArn);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        roleArn = createSageMakerRole(iam, roleName);
    }
    return roleArn;
}

private static String[] getSageMakerRolePolicies() {
    String[] sageMakerRolePolicies = new String[3];
    sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
    sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/" +
    "AmazonSageMakerGeospatialFullAccess";
    sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
    return sageMakerRolePolicies;
}

private static String[] getLambdaRolePolicies() {
    String[] lambdaRolePolicies = new String[5];
    lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
    lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
    lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
    "AmazonSageMakerGeospatialFullAccess";
    lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/"
        + "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy";
    lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
    "AWSLambdaSQSQueueExecutionRole";
    return lambdaRolePolicies;
}
}

```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [CreatePipeline](#)
  - [DeletePipeline](#)
  - [DescribePipelineExecution](#)
  - [StartPipelineExecution](#)

- [UpdatePipeline](#)

## Secrets Manager contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan Secrets Manager AWS SDK for Java 2.x with.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

### Tindakan

Buat rahasia

Contoh kode berikut menunjukkan cara membuat rahasia Secrets Manager.

SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.CreateSecretRequest;
import software.amazon.awssdk.services.secretsmanager.model.CreateSecretResponse;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateSecret {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <secretName> <secretValue>\s

            Where:
                secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                secretValue - The secret value.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String secretName = args[0];
        String secretValue = args[1];
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
            .region(region)
            .build();

        String secretARN = createNewSecret(secretsClient, secretName, secretValue);
        System.out.println("The secret ARN is " + secretARN);
        secretsClient.close();
    }

    public static String createNewSecret(SecretsManagerClient secretsClient, String
secretName, String secretValue) {
        try {
            CreateSecretRequest secretRequest = CreateSecretRequest.builder()
                .name(secretName)
```

```

        .description("This secret was created by the AWS Secret Manager
Java API")
        .secretString(secretValue)
        .build();

    CreateSecretResponse secretResponse =
secretsClient.createSecret(secretRequest);
    return secretResponse.arn();

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
}

```

- Untuk detail API, lihat [CreateSecret](#) di Referensi AWS SDK for Java 2.x API.

## Hapus rahasia

Contoh kode berikut menunjukkan cara menghapus rahasia Secrets Manager.

### SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.DeleteSecretRequest;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:

```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DeleteSecret {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <secretName>\s

            Where:
                secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String secretName = args[0];
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
            .region(region)
            .build();

        deleteSpecificSecret(secretsClient, secretName);
        secretsClient.close();
    }

    public static void deleteSpecificSecret(SecretsManagerClient secretsClient,
String secretName) {
        try {
            DeleteSecretRequest secretRequest = DeleteSecretRequest.builder()
                .secretId(secretName)
                .build();

            secretsClient.deleteSecret(secretRequest);
            System.out.println(secretName + " is deleted.");

        } catch (SecretsManagerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```




```
    }  
  }  
}
```

- Untuk detail API, lihat [DeleteSecret](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan sebuah rahasia

Contoh kode berikut menunjukkan cara mendeskripsikan rahasia Secrets Manager.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;  
import software.amazon.awssdk.services.secretsmanager.model.DescribeSecretRequest;  
import software.amazon.awssdk.services.secretsmanager.model.DescribeSecretResponse;  
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;  
import java.time.Instant;  
import java.time.ZoneId;  
import java.time.format.DateTimeFormatter;  
import java.time.format.FormatStyle;  
import java.util.Locale;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class DescribeSecret {  
    public static void main(String[] args) {  
        final String usage = ""
```

```
Usage:
    <secretName>\s

Where:
    secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
    """";

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String secretName = args[0];
Region region = Region.US_EAST_1;
SecretsManagerClient secretsClient = SecretsManagerClient.builder()
    .region(region)
    .build();

describeGivenSecret(secretsClient, secretName);
secretsClient.close();
}

public static void describeGivenSecret(SecretsManagerClient secretsClient,
String secretName) {
    try {
        DescribeSecretRequest secretRequest = DescribeSecretRequest.builder()
            .secretId(secretName)
            .build();

        DescribeSecretResponse secretResponse =
secretsClient.describeSecret(secretRequest);
        Instant lastChangedDate = secretResponse.lastChangedDate();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(lastChangedDate);
        System.out.println("The date of the last change to " +
secretResponse.name() + " is " + lastChangedDate);
    }
}
```

```
    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DescribeSecret](#) di Referensi AWS SDK for Java 2.x API.

## Dapatkan nilai rahasia

Contoh kode berikut menunjukkan cara mendapatkan nilai rahasia Secrets Manager.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * We recommend that you cache your secret values by using client-side caching.
 *
 * Caching secrets improves speed and reduces your costs. For more information,
 * see the following documentation topic:
 *
 */
```

```
* https://docs.aws.amazon.com/secretsmanager/latest/userguide/retrieving-secrets.html
*/
public class GetSecretValue {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <secretName>\s

            Where:
                secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String secretName = args[0];
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
            .region(region)
            .build();

        getValue(secretsClient, secretName);
        secretsClient.close();
    }

    public static void getValue(SecretsManagerClient secretsClient, String
secretName) {
        try {
            GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

            GetSecretValueResponse valueResponse =
secretsClient.getSecretValue(valueRequest);
            String secret = valueResponse.secretString();
            System.out.println(secret);

        } catch (SecretsManagerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

```
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [GetSecretValue](#) di Referensi AWS SDK for Java 2.x API.

## Daftar rahasia

Contoh kode berikut menunjukkan cara daftar rahasia Secrets Manager.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.ListSecretsResponse;
import software.amazon.awssdk.services.secretsmanager.model.SecretListEntry;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListSecrets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
            .region(region)
            .build();
    }
}
```

```

        listAllSecrets(secretsClient);
        secretsClient.close();
    }

    public static void listAllSecrets(SecretsManagerClient secretsClient) {
        try {
            ListSecretsResponse secretsResponse = secretsClient.listSecrets();
            List<SecretListEntry> secrets = secretsResponse.secretList();
            for (SecretListEntry secret : secrets) {
                System.out.println("The secret name is " + secret.name());
                System.out.println("The secret description is " +
secret.description());
            }

        } catch (SecretsManagerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Untuk detail API, lihat [ListSecrets](#) di Referensi AWS SDK for Java 2.x API.

## Memodifikasi detail rahasia

Contoh kode berikut menunjukkan cara memodifikasi rahasia.

SDK for Java 2.x

### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;
import software.amazon.awssdk.services.secretsmanager.model.UpdateSecretRequest;

/**

```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class UpdateSecret {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <secretName> <secretValue>

            Where:
                secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                secretValue - The secret value that is updated.\s
            """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String secretName = args[0];
        String secretValue = args[1];
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
            .region(region)
            .build();

        updateMySecret(secretsClient, secretName, secretValue);
        secretsClient.close();
    }

    public static void updateMySecret(SecretsManagerClient secretsClient, String
secretName, String secretValue) {
        try {
            UpdateSecretRequest secretRequest = UpdateSecretRequest.builder()
                .secretId(secretName)
                .secretString(secretValue)
                .build();
```

```

        secretsClient.updateSecret(secretRequest);

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [UpdateSecret](#) di Referensi AWS SDK for Java 2.x API.

## Menempatkan nilai dalam rahasia

Contoh kode berikut menunjukkan cara menempatkan nilai dalam rahasia Secrets Manager.

SDK for Java 2.x

### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.PutSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutSecret {
    public static void main(String[] args) {
        final String usage = ""

        Usage:

```



```
        <secretName> <secretValue>

        Where:
            secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
            secretValue - The text to encrypt and store in the new version
of the secret.\s
            """;

    if (args.length < 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String secretName = args[0];
    String secretValue = args[1];
    Region region = Region.US_EAST_1;
    SecretsManagerClient secretsClient = SecretsManagerClient.builder()
        .region(region)
        .build();

    putSecret(secretsClient, secretName, secretValue);
    secretsClient.close();
}

public static void putSecret(SecretsManagerClient secretsClient, String
secretName, String secretValue) {
    try {
        PutSecretValueRequest secretRequest = PutSecretValueRequest.builder()
            .secretId(secretName)
            .secretString(secretValue)
            .build();

        secretsClient.putSecretValue(secretRequest);
        System.out.println("A new version was created.");

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [PutSecretValue](#) di Referensi AWS SDK for Java 2.x API.

## Amazon SES contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x With Amazon SES.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

### Tindakan

Daftar template email

Contoh kode berikut menunjukkan cara membuat daftar template email Amazon SES.

SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesRequest;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesResponse;
```

```
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;

public class ListTemplates {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
            .region(region)
            .build();

        listAllTemplates(sesv2Client);
    }

    public static void listAllTemplates(SesV2Client sesv2Client) {
        try {
            ListEmailTemplatesRequest templatesRequest =
                ListEmailTemplatesRequest.builder()
                    .pageSize(1)
                    .build();


            ListEmailTemplatesResponse response =
                sesv2Client.listEmailTemplates(templatesRequest);
            response.templatesMetadata()
                .forEach(template -> System.out.println("Template name: " +
                    template.templateName()));
        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListTemplates](#) di Referensi AWS SDK for Java 2.x API.

## Daftar identitas

Contoh kode berikut menunjukkan cara mencantumkan identitas Amazon SES.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.ListIdentitiesResponse;
import software.amazon.awssdk.services.ses.model.SesException;
import java.io.IOException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentities {

    public static void main(String[] args) throws IOException {
        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
            .region(region)
            .build();

        listSESIIdentities(client);
    }

    public static void listSESIIdentities(SesClient client) {
        try {
            ListIdentitiesResponse identitiesResponse = client.listIdentities();
            List<String> identities = identitiesResponse.identities();
            for (String identity : identities) {
                System.out.println("The identity is " + identity);
            }
        }
    }
}
```

```
        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListIdentities](#) di Referensi AWS SDK for Java 2.x API.

## Kirim email

Contoh kode berikut menunjukkan cara mengirim email dengan Amazon SES.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.Content;
import software.amazon.awssdk.services.ses.model.Destination;
import software.amazon.awssdk.services.ses.model.Message;
import software.amazon.awssdk.services.ses.model.Body;
import software.amazon.awssdk.services.ses.model.SendEmailRequest;
import software.amazon.awssdk.services.ses.model.SesException;

import javax.mail.MessagingException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageEmailRequest {
```

```
public static void main(String[] args) {
    final String usage = ""

        Usage:
            <sender> <recipient> <subject>\s

        Where:
            sender - An email address that represents the sender.\s
            recipient - An email address that represents the recipient.\s
            subject - The subject line.\s
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String sender = args[0];
    String recipient = args[1];
    String subject = args[2];

    Region region = Region.US_EAST_1;
    SesClient client = SesClient.builder()
        .region(region)
        .build();

    // The HTML body of the email.
    String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
        + "<p> See the list of customers.</p>" + "</body>" + "</html>";

    try {
        send(client, sender, recipient, subject, bodyHTML);
        client.close();
        System.out.println("Done");
    } catch (MessagingException e) {
        e.printStackTrace();
    }
}

public static void send(SesClient client,
    String sender,
    String recipient,
    String subject,
```

```
String bodyHTML) throws MessagingException {

    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    Content content = Content.builder()
        .data(bodyHTML)
        .build();

    Content sub = Content.builder()
        .data(subject)
        .build();

    Body body = Body.builder()
        .html(content)
        .build();

    Message msg = Message.builder()
        .subject(sub)
        .body(body)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
        .message(msg)
        .source(sender)
        .build();

    try {
        System.out.println("Attempting to send an email through Amazon SES " +
            "using the AWS SDK for Java...");
        client.sendEmail(emailRequest);
    } catch (SesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import javax.activation.DataHandler;
```

```
import javax.activation.DataSource;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.mail.internet.MimeBodyPart;
import javax.mail.util.ByteArrayDataSource;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.file.Files;
import java.util.Properties;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.ses.model.SendRawEmailRequest;
import software.amazon.awssdk.services.ses.model.RawMessage;
import software.amazon.awssdk.services.ses.model.SesException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SendMessageAttachment {
    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <sender> <recipient> <subject> <fileLocation>\s

            Where:
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
                subject - The subject line.\s
                fileLocation - The location of a Microsoft Excel file to use as
                an attachment (C:/AWS/customers.xls).\s
            """;
```



```
    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String sender = args[0];
    String recipient = args[1];
    String subject = args[2];
    String fileLocation = args[3];

    // The email body for recipients with non-HTML email clients.
    String bodyText = "Hello,\r\n" + "Please see the attached file for a list "
        + "of customers to contact.";

    // The HTML body of the email.
    String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
        + "<p>Please see the attached file for a " + "list of customers to
contact.</p>" + "</body>"
        + "</html>";

    Region region = Region.US_WEST_2;
    SesClient client = SesClient.builder()
        .region(region)
        .build();

    try {
        sendemailAttachment(client, sender, recipient, subject, bodyText,
bodyHTML, fileLocation);
        client.close();
        System.out.println("Done");
    } catch (IOException | MessagingException e) {
        e.printStackTrace();
    }
}

public static void sendemailAttachment(SesClient client,
    String sender,
    String recipient,
    String subject,
    String bodyText,
    String bodyHTML,
    String fileLocation) throws AddressException, MessagingException,
IOException {
```

```
java.io.File theFile = new java.io.File(fileLocation);
byte[] fileContent = Files.readAllBytes(theFile.toPath());

Session session = Session.getDefaultInstance(new Properties());

// Create a new MimeMessage object.
MimeMessage message = new MimeMessage(session);

// Add subject, from and to lines.
message.setSubject(subject, "UTF-8");
message.setFrom(new InternetAddress(sender));
message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(recipient));

// Create a multipart/alternative child container.
MimeMultipart msgBody = new MimeMultipart("alternative");

// Create a wrapper for the HTML and text parts.
MimeBodyPart wrap = new MimeBodyPart();

// Define the text part.
MimeBodyPart textPart = new MimeBodyPart();
textPart.setContent(bodyText, "text/plain; charset=UTF-8");

// Define the HTML part.
MimeBodyPart htmlPart = new MimeBodyPart();
htmlPart.setContent(bodyHTML, "text/html; charset=UTF-8");

// Add the text and HTML parts to the child container.
msgBody.addBodyPart(textPart);
msgBody.addBodyPart(htmlPart);

// Add the child container to the wrapper object.
wrap.setContent(msgBody);

// Create a multipart/mixed parent container.
MimeMultipart msg = new MimeMultipart("mixed");

// Add the parent container to the message.
message.setContent(msg);
msg.addBodyPart(wrap);

// Define the attachment.
```

```
MimeBodyPart att = new MimeBodyPart();
DataSource fds = new ByteArrayDataSource(fileContent,
    "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet");
att.setDataHandler(new DataHandler(fds));

String reportName = "WorkReport.xls";
att.setFileName(reportName);

// Add the attachment to the message.
msg.addBodyPart(att);

try {
    System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");

    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    message.writeTo(outputStream);

    ByteBuffer buf = ByteBuffer.wrap(outputStream.toByteArray());

    byte[] arr = new byte[buf.remaining()];
    buf.get(arr);

    SdkBytes data = SdkBytes.fromByteArray(arr);
    RawMessage rawMessage = RawMessage.builder()
        .data(data)
        .build();

    SendRawEmailRequest rawEmailRequest = SendRawEmailRequest.builder()
        .rawMessage(rawMessage)
        .build();

    client.sendRawEmail(rawEmailRequest);

} catch (SesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.println("Email sent using SesClient with attachment");
}
```

- Untuk detail API, lihat [SendEmail](#) di Referensi AWS SDK for Java 2.x API.

## Kirim email template

Contoh kode berikut menunjukkan cara mengirim email template dengan Amazon SES.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.Template;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Also, make sure that you create a template. See the following documentation
 * topic:
 *
 * https://docs.aws.amazon.com/ses/latest/dg/send-personalized-email-api.html
 */

public class SendEmailTemplate {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <template> <sender> <recipient>\s
```

```

        Where:
            template - The name of the email template.
            sender - An email address that represents the sender.\s
            recipient - An email address that represents the recipient.\s
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String templateName = args[0];
    String sender = args[1];
    String recipient = args[2];
    Region region = Region.US_EAST_1;
    SesV2Client sesv2Client = SesV2Client.builder()
        .region(region)
        .build();

    send(sesv2Client, sender, recipient, templateName);
}

public static void send(SesV2Client client, String sender, String recipient,
String templateName) {
    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    /*
     * Specify both name and favorite animal (favoriteanimal) in your code when
     * defining the Template object.
     * If you don't specify all the variables in the template, Amazon SES
doesn't
     * send the email.
     */
    Template myTemplate = Template.builder()
        .templateName(templateName)
        .templateData("{\n" +
            "  \"name\": \"Jason\"\n," +
            "  \"favoriteanimal\": \"Cat\"\n" +
            "}")
        .build();

```

```
    EmailContent emailContent = EmailContent.builder()
        .template(myTemplate)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
        .content(emailContent)
        .fromEmailAddress(sender)
        .build();

    try {
        System.out.println("Attempting to send an email based on a template
using the AWS SDK for Java (v2)...");
        client.sendEmail(emailRequest);
        System.out.println("email based on a template was sent");

    } catch (SesV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [SendTemplatedEmail](#) di Referensi AWS SDK for Java 2.x API.

## Amazon SES API v2 contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with Amazon SES API v2.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

### Topik

- [Tindakan](#)

## Tindakan

### Kirim email

Contoh kode berikut menunjukkan cara mengirim email Amazon SES API v2.

### SDK for Java 2.x

#### Note

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

### Mengirim pesan.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Body;
import software.amazon.awssdk.services.sesv2.model.Content;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.Message;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SendEmail {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
```

```

        <sender> <recipient> <subject>\s

Where:
    sender - An email address that represents the
sender.\s
    recipient - An email address that represents the
recipient.\s
    subject - The subject line.\s
""";

if (args.length != 3) {
    System.out.println(usage);
    System.exit(1);
}

String sender = args[0];
String recipient = args[1];
String subject = args[2];

Region region = Region.US_EAST_1;
SesV2Client sesv2Client = SesV2Client.builder()
    .region(region)
    .build();

// The HTML body of the email.
String bodyHTML = "<html>" + "<head></head>" + "<body>" +
"<h1>Hello!</h1>"
    + "<p> See the list of customers.</p>" + "</body>" +
"</html>";

    send(sesv2Client, sender, recipient, subject, bodyHTML);
}

public static void send(SesV2Client client,
    String sender,
    String recipient,
    String subject,
    String bodyHTML) {

    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    Content content = Content.builder()

```



```
        .data(bodyHTML)
        .build();

    Content sub = Content.builder()
        .data(subject)
        .build();

    Body body = Body.builder()
        .html(content)
        .build();

    Message msg = Message.builder()
        .subject(sub)
        .body(body)
        .build();

    EmailContent emailContent = EmailContent.builder()
        .simple(msg)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
        .content(emailContent)
        .fromEmailAddress(sender)
        .build();

    try {
        System.out.println("Attempting to send an email through
Amazon SES "
            + "using the AWS SDK for Java...");
        client.sendEmail(emailRequest);
        System.out.println("email was sent");

    } catch (SesV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [SendEmail](#) di Referensi AWS SDK for Java 2.x API.

## Contoh Amazon SNS menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK for Java 2.x dengan Amazon SNS.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Memulai

Halo Amazon SNS

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon SNS.

SDK for Java 2.x

### Note

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

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
}
```

```
        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                .flatMap(r -> r.topics().stream())
                .forEach(content -> System.out.println(" Topic ARN: " +
content.topicArn()));

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListTopics](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

### Menambahkan tag ke topik

Contoh kode berikut menunjukkan cara menambahkan tag ke topik Amazon SNS.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddTags {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn>

            Where:
                topicArn - The ARN of the topic to which tags are added.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        addTopicTags(snsClient, topicArn);
        snsClient.close();
    }

    public static void addTopicTags(SnsClient snsClient, String topicArn) {
        try {
```

```
        Tag tag = Tag.builder()
            .key("Team")
            .value("Development")
            .build();

        Tag tag2 = Tag.builder()
            .key("Environment")
            .value("Gamma")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag);
        tagList.add(tag2);

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .resourceArn(topicArn)
            .tags(tagList)
            .build();

        snsClient.tagResource(tagResourceRequest);
        System.out.println("Tags have been added to " + topicArn);

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [TagResource](#) di Referensi AWS SDK for Java 2.x API.

Periksa apakah ada nomor telepon yang dipilih keluar

Contoh kode berikut menunjukkan cara memeriksa apakah nomor telepon dipilih untuk tidak menerima pesan Amazon SNS.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
    software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
    software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CheckOptOut {
    public static void main(String[] args) {

        final String usage = ""

            Usage:    <phoneNumber>

            Where:
                phoneNumber - The mobile phone number to look up (for example,
+1XXX5550100).

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String phoneNumber = args[0];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .build();

checkPhone(snsClient, phoneNumber);
snsClient.close();
}

public static void checkPhone(SnsClient snsClient, String phoneNumber) {
    try {
        CheckIfPhoneNumberIsOptedOutRequest request =
        CheckIfPhoneNumberIsOptedOutRequest.builder()
            .phoneNumber(phoneNumber)
            .build();

        CheckIfPhoneNumberIsOptedOutResponse result =
        snsClient.checkIfPhoneNumberIsOptedOut(request);
        System.out.println(
            result.isOptedOut() + "Phone Number " + phoneNumber + " has
Opted Out of receiving sns messages." +
            "\n\nStatus was " +
            result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [CheckIfPhoneNumbersIsOptedOut](#) di Referensi AWS SDK for Java 2.x API.

Konfirmasikan pemilik endpoint ingin menerima pesan

Contoh kode berikut menunjukkan cara mengonfirmasi pemilik titik akhir ingin menerima pesan Amazon SNS dengan memvalidasi token yang dikirim ke titik akhir dengan tindakan Berlangganan sebelumnya.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionRequest;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ConfirmSubscription {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <subscriptionToken> <topicArn>

                Where:
                    subscriptionToken - A short-lived token sent to an endpoint
during the Subscribe action.
                    topicArn - The ARN of the topic.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionToken = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
```



```
        .region(Region.US_EAST_1)
        .build();

    confirmSub(snsClient, subscriptionToken, topicArn);
    snsClient.close();
}

public static void confirmSub(SnsClient snsClient, String subscriptionToken,
String topicArn) {
    try {
        ConfirmSubscriptionRequest request =
ConfirmSubscriptionRequest.builder()
            .token(subscriptionToken)
            .topicArn(topicArn)
            .build();

        ConfirmSubscriptionResponse result =
snsClient.confirmSubscription(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n"
            + result.subscriptionArn());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [ConfirmSubscription](#) di Referensi AWS SDK for Java 2.x API.

## Buat topik

Contoh kode berikut menunjukkan cara membuat topik Amazon SNS.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicName>

            Where:
                topicName - The name of the topic to create (for example,
mytopic).

            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
```

```
    CreateTopicResponse result;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Untuk detail API, lihat [CreateTopic](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus langganan

Contoh kode berikut menunjukkan cara menghapus langganan Amazon SNS.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <subscriptionArn>

            Where:
                subscriptionArn - The ARN of the subscription to delete.
            """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        unSub(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode()
                + "\n\nSubscription was removed for " +
request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
}  
}
```

- Untuk detail API, lihat [Berhenti berlangganan](#) di Referensi AWS SDK for Java 2.x API.

## Hapus topik

Contoh kode berikut menunjukkan cara menghapus topik Amazon SNS dan semua langganan ke topik itu.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sns.SnsClient;  
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;  
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;  
import software.amazon.awssdk.services.sns.model.SnsException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class DeleteTopic {  
    public static void main(String[] args) {  
        final String usage = ""  
  
            Usage:    <topicArn>  
  
            Where:  
                topicArn - The ARN of the topic to delete.  
    }  
}
```

```
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String topicArn = args[0];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    System.out.println("Deleting a topic with name: " + topicArn);
    deleteSNSTopic(snsClient, topicArn);
    snsClient.close();
}

public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
    try {
        DeleteTopicRequest request = DeleteTopicRequest.builder()
            .topicArn(topicArn)
            .build();

        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DeleteTopic](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan properti dari suatu topik

Contoh kode berikut menunjukkan cara mendapatkan properti topik Amazon SNS.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetTopicAttributes {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn>

            Where:
                topicArn - The ARN of the topic to look up.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
```

```

        System.out.println("Getting attributes for a topic with name: " + topicArn);
        getSNSTopicAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSTopicAttributes(SnsClient snsClient, String topicArn) {
        try {
            GetTopicAttributesRequest request = GetTopicAttributesRequest.builder()
                .topicArn(topicArn)
                .build();

            GetTopicAttributesResponse result =
snsClient.getTopicAttributes(request);
            System.out.println("\n\nStatus is " +
result.sdkHttpResponse().statusCode() + "\n\nAttributes: \n\n"
                + result.attributes());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Untuk detail API, lihat [GetTopicAttributes](#) di Referensi AWS SDK for Java 2.x API.

## Dapatkan pengaturan untuk mengirim pesan SMS

Contoh kode berikut menunjukkan cara mendapatkan pengaturan untuk mengirim pesan SMS Amazon SNS.

### SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;

```



```
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetSMSAttributes {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <topicArn>

                Where:
                    topicArn - The ARN of the topic from which to retrieve
attributes.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        getSnsAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSnsAttributes(SnsClient snsClient, String topicArn) {
        try {
            GetSubscriptionAttributesRequest request =
GetSubscriptionAttributesRequest.builder()
                .subscriptionArn(topicArn)
```

```
        .build();

        // Get the Subscription attributes
        GetSubscriptionAttributesResponse res =
snsClient.getSubscriptionAttributes(request);
        Map<String, String> map = res.attributes();

        // Iterate through the map
        Iterator iter = map.entrySet().iterator();
        while (iter.hasNext()) {
            Map.Entry entry = (Map.Entry) iter.next();
            System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
entry.getValue());
        }

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    System.out.println("\n\nStatus was good");
}
}
```

- Untuk detail API, lihat [GetSmSatTributes](#) di Referensi API.AWS SDK for Java 2.x

Daftar nomor telepon yang dipilih keluar

Contoh kode berikut menunjukkan cara mencantumkan nomor telepon yang dipilih untuk tidak menerima pesan Amazon SNS.

SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " + result.sdkHttpResponse().statusCode())
+ "\n\nPhone Numbers: \n\n"
                + result.phoneNumbers());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListPhoneNumbersOptedOut](#) di Referensi AWS SDK for Java 2.x API.

## Buat daftar pelanggan suatu topik

Contoh kode berikut menunjukkan cara mengambil daftar pelanggan topik Amazon SNS.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsRequest;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListSubscriptions {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listSNSSubscriptions(snsClient);
        snsClient.close();
    }

    public static void listSNSSubscriptions(SnsClient snsClient) {
        try {
            ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
                .build();

            ListSubscriptionsResponse result = snsClient.listSubscriptions(request);
            System.out.println(result.subscriptions());
        }
    }
}
```

```
        } catch (SnsException e) {  
  
            System.err.println(e.awsErrorDetails().errorMessage());  
            System.exit(1);  
        }  
    }  
}
```

- Untuk detail API, lihat [ListSubscriptions](#) di Referensi AWS SDK for Java 2.x API.

## Daftar topik

Contoh kode berikut menunjukkan cara membuat daftar topik Amazon SNS.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sns.SnsClient;  
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;  
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;  
import software.amazon.awssdk.services.sns.model.SnsException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class ListTopics {  
    public static void main(String[] args) {  
        SnsClient snsClient = SnsClient.builder()  
            .region(Region.US_EAST_1)
```

```
        .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
                .build();

            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println(
                "Status was " + result.sdkHttpResponse().statusCode() + "\n\n"
                + result.topics());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListTopics](#) di Referensi AWS SDK for Java 2.x API.

## Publikasikan pesan teks SMS

Contoh kode berikut menunjukkan cara mempublikasikan pesan SMS menggunakan Amazon SNS.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
```

```
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <message> <phoneNumber>

                Where:
                    message - The message text to send.
                    phoneNumber - The mobile phone number to which a message is sent
(for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .phoneNumber(phoneNumber)
                .build();
        }
    }
}
```

```
        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [Publikasikan](#) di Referensi AWS SDK for Java 2.x API.

## Publikasikan ke topik

Contoh kode berikut menunjukkan cara mempublikasikan pesan ke topik Amazon SNS.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTopic {
```



```
public static void main(String[] args) {
    final String usage = ""

        Usage:    <message> <topicArn>

        Where:
            message - The message text to send.
            topicArn - The ARN of the topic to publish.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String message = args[0];
    String topicArn = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();
    pubTopic(snsClient, message, topicArn);
    snsClient.close();
}

public static void pubTopic(SnsClient snsClient, String message, String
topicArn) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .topicArn(topicArn)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [Publikasikan](#) di Referensi AWS SDK for Java 2.x API.

## Menetapkan kebijakan filter

Contoh kode berikut menunjukkan cara menyetel kebijakan filter Amazon SNS.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <subscriptionArn>

                Where:
                    subscriptionArn - The ARN of a subscription.

                """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String subscriptionArn = args[0];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    usePolicy(snsClient, subscriptionArn);
    snsClient.close();
}

public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
    try {
        SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
        // Add a filter policy attribute with a single value
        fp.addAttribute("store", "example_corp");
        fp.addAttribute("event", "order_placed");

        // Add a prefix attribute
        fp.addAttributePrefix("customer_interests", "bas");

        // Add an anything-but attribute
        fp.addAttributeAnythingBut("customer_interests", "baseball");

        // Add a filter policy attribute with a list of values
        ArrayList<String> attributeValues = new ArrayList<>();
        attributeValues.add("rugby");
        attributeValues.add("soccer");
        attributeValues.add("hockey");
        fp.addAttribute("customer_interests", attributeValues);

        // Add a numeric attribute
        fp.addAttribute("price_usd", "=", 0);

        // Add a numeric attribute with a range
        fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

        // Apply the filter policy attributes to an Amazon SNS subscription
        fp.apply(snsClient, subscriptionArn);

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
}
```

- Untuk detail API, lihat [SetSubscriptionAttributes](#) di Referensi AWS SDK for Java 2.x API.

## Atur pengaturan default untuk mengirim pesan SMS

Contoh kode berikut menunjukkan cara mengatur pengaturan default untuk mengirim pesan SMS menggunakan Amazon SNS.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sns.SnsClient;  
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;  
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;  
import software.amazon.awssdk.services.sns.model.SnsException;  
import java.util.HashMap;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class SetSMSAttributes {  
    public static void main(String[] args) {  
        HashMap<String, String> attributes = new HashMap<>(1);  
        attributes.put("DefaultSMSType", "Transactional");  
        attributes.put("UsageReportS3Bucket", "janbucket");  
  
        SnsClient snsClient = SnsClient.builder()
```

```

        .region(Region.US_EAST_1)
        .build();
    setSNSAttributes(snsClient, attributes);
    snsClient.close();
}

public static void setSNSAttributes(SnsClient snsClient, HashMap<String, String>
attributes) {
    try {
        SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
            .attributes(attributes)
            .build();

        SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
        System.out.println("Set default Attributes to " + attributes + ". Status
was "
            + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [SetSmSatTributes](#) di Referensi API.AWS SDK for Java 2.x

Tetapkan atribut topik

Contoh kode berikut menunjukkan cara mengatur atribut topik Amazon SNS.

SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;

```

```
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <attribute> <topicArn> <value>

            Where:
                attribute - The attribute action to use. Valid parameters are:
Policy | DisplayName | DeliveryPolicy .
                topicArn - The ARN of the topic.\s
                value - The value for the attribute.
            """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        setTopAttr(snsClient, attribute, topicArn, value);
        snsClient.close();
    }
}
```

```
public static void setTopAttr(SnsClient snsClient, String attribute, String
topicArn, String value) {
    try {
        SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
            .attributeName(attribute)
            .attributeValue(value)
            .topicArn(topicArn)
            .build();

        SetTopicAttributesResponse result =
snsClient.setTopicAttributes(request);
        System.out.println(
            "\n\nStatus was " + result.sdkHttpResponse().statusCode() + "\n
\nTopic " + request.topicArn()
                + " updated " + request.attributeName() + " to " +
request.attributeValue());


    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [SetTopicAttributes](#) di Referensi AWS SDK for Java 2.x API.

Berlangganan fungsi Lambda ke suatu topik

Contoh kode berikut menunjukkan cara berlangganan fungsi Lambda sehingga menerima pemberitahuan dari topik Amazon SNS.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
```

```
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <topicArn> <lambdaArn>

            Where:
                topicArn - The ARN of the topic to subscribe.
                lambdaArn - The ARN of an AWS Lambda function.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }

    public static String subLambda(SnsClient snsClient, String topicArn, String
lambdaArn) {
        try {
```



```
SubscribeRequest request = SubscribeRequest.builder()
    .protocol("lambda")
    .endpoint(lambdaArn)
    .returnSubscriptionArn(true)
    .topicArn(topicArn)
    .build();

SubscribeResponse result = snsClient.subscribe(request);
return result.subscriptionArn();


} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
}
```

- Untuk detail API, lihat [Berlangganan](#) di Referensi AWS SDK for Java 2.x API.

Berlangganan titik akhir HTTP ke suatu topik

Contoh kode berikut menunjukkan cara berlangganan titik akhir HTTP atau HTTPS sehingga menerima pemberitahuan dari topik Amazon SNS.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn> <url>

            Where:
                topicArn - The ARN of the topic to subscribe.
                url - The HTTPS endpoint that you want to receive notifications.
            """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String url = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        subHTTPS(snsClient, topicArn, url);
        snsClient.close();
    }

    public static void subHTTPS(SnsClient snsClient, String topicArn, String url) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("https")
                .endpoint(url)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN is " + result.subscriptionArn() +
                "\n\n Status is ")
        }
    }
}
```

```

        + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- Untuk detail API, lihat [Berlangganan](#) di Referensi AWS SDK for Java 2.x API.

Berlangganan alamat email ke suatu topik

Contoh kode berikut menunjukkan cara berlangganan alamat email ke topik Amazon SNS.

SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeEmail {
    public static void main(String[] args) {
        final String usage = ""
            Usage:    <topicArn> <email>

```

```
        Where:
            topicArn - The ARN of the topic to subscribe.
            email - The email address to use.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String topicArn = args[0];
    String email = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    subEmail(snsClient, topicArn, email);
    snsClient.close();
}

public static void subEmail(SnsClient snsClient, String topicArn, String email)
{
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("email")
            .endpoint(email)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
            + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [Berlangganan](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

Buat titik akhir platform untuk pemberitahuan push

Contoh kode berikut menunjukkan cara membuat titik akhir platform untuk notifikasi push Amazon SNS.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
```

```
final String usage = ""

    Usage:    <token> <platformApplicationArn>

    Where:
        token - The name of the FIFO topic.\s
        platformApplicationArn - The ARN value of platform application.
You can get this value from the AWS Management Console.\s
    """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String token = args[0];
    String platformApplicationArn = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    createEndpoint(snsClient, token, platformApplicationArn);
}


public static void createEndpoint(SnsClient snsClient, String token, String
platformApplicationArn) {
    System.out.println("Creating platform endpoint with token " + token);
    try {
        CreatePlatformEndpointRequest endpointRequest =
CreatePlatformEndpointRequest.builder()
            .token(token)
            .platformApplicationArn(platformApplicationArn)
            .build();

        CreatePlatformEndpointResponse response =
snsClient.createPlatformEndpoint(endpointRequest);
        System.out.println("The ARN of the endpoint is " +
response.endpointArn());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

Buat dan publikasikan ke topik FIFO

Contoh kode berikut menunjukkan cara membuat dan mempublikasikan ke topik FIFO Amazon SNS.

SDK for Java 2.x

 Note

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

Contoh ini

- membuat topik Amazon SNS FIFO, dua antrian FIFO Amazon SQS, dan satu antrian Standar.
- berlangganan antrian ke topik dan menerbitkan pesan ke topik tersebut.

[Tes](#) memverifikasi penerimaan pesan ke setiap antrian. [Contoh lengkap](#) juga menunjukkan penambahan kebijakan akses dan menghapus sumber daya di akhir.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
            "Where:\n" +
            "    fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
            "    wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
            +
            "    retailQueueARN - The name of a SQS FIFO queue that will created
for the retail consumer. \n\n" +
            "    analyticsQueueARN - The name of a SQS standard queue that will
be created for the analytics consumer. \n\n";
```

```
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue: ARN,
URL,
        // name and type.
        List<QueueData> queues = List.of(
            new QueueData(wholeSaleQueueName, QueueType.FIFO),
            new QueueData(retailQueueName, QueueType.FIFO),
            new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
```



```

        "FifoTopic", "true",
        "ContentBasedDeduplication", "false");

    CreateTopicRequest topicRequest = CreateTopicRequest.builder()
        .name(topicName)
        .attributes(topicAttributes)
        .build();

    CreateTopicResponse response = snsClient.createTopic(topicRequest);
    String topicArn = response.topicArn();
    System.out.println("The topic ARN is" + topicArn);

    return topicArn;

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static void subscribeQueues(List<QueueData> queues, String topicARN) {
    queues.forEach(queue -> {
        SubscribeRequest subscribeRequest = SubscribeRequest.builder()
            .topicArn(topicARN)
            .endpoint(queue.queueARN)
            .protocol("sqs")
            .build();

        // Subscribe to the endpoint by using the SNS service client.
        // Only Amazon SQS queues can receive notifications from an Amazon SNS
        FIFO
        // topic.
        SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
        System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]);
        queue.subscriptionARN = subscribeResponse.subscriptionArn();
    });
}

public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

```

```
try {
    // Create and publish a message that updates the wholesale price.
    String subject = "Price Update";
    String dedupId = UUID.randomUUID().toString();
    String attributeName = "business";
    String attributeValue = "wholesale";

    MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
        .dataType("String")
        .stringValue(attributeValue)
        .build();

    Map<String, MessageAttributeValue> attributes = new HashMap<>();
    attributes.put(attributeName, msgAttValue);
    PublishRequest pubRequest = PublishRequest.builder()
        .topicArn(topicArn)
        .subject(subject)
        .message(payload)
        .messageGroupId(groupId)
        .messageDeduplicationId(dedupId)
        .messageAttributes(attributes)
        .build();

    final PublishResponse response = snsClient.publish(pubRequest);
    System.out.println(response.messageId());
    System.out.println(response.sequenceNumber());
    System.out.println("Message was published to " + topicArn);

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .

- [CreateTopic](#)
- [Publikasikan](#)
- [Berlangganan](#)

## Publikasikan pesan SMS ke suatu topik

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat topik Amazon SNS.
- Berlangganan nomor telepon ke topik.
- Publikasikan pesan SMS ke topik sehingga semua nomor telepon berlangganan menerima pesan sekaligus.

## SDK for Java 2.x

### Note

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

Buat topik dan kembalikan ARN-nya.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <topicName>

                Where:
```

```

        topicName - The name of the topic to create (for example,
mytopic).

        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String topicName = args[0];
    System.out.println("Creating a topic with name: " + topicName);
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    String arnVal = createSNSTopic(snsClient, topicName);
    System.out.println("The topic ARN is" + arnVal);
    snsClient.close();
}

public static String createSNSTopic(SnsClient snsClient, String topicName) {
    CreateTopicResponse result;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}

```

Berlangganan titik akhir ke suatu topik.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <topicArn> <phoneNumber>

                Where:
                    topicArn - The ARN of the topic to subscribe.
                    phoneNumber - A mobile phone number that receives notifications
(for example, +1XXX5550100).
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
```

```

        .protocol("sms")
        .endpoint(phoneNumber)
        .returnSubscriptionArn(true)
        .topicArn(topicArn)
        .build();

    SubscribeResponse result = snsClient.subscribe(request);
    System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
        + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

Tetapkan atribut pada pesan, seperti ID pengirim, harga maksimum, dan jenisnya. Atribut pesan bersifat opsional.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");
    }
}

```

```

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String, String>
attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                .attributes(attributes)
                .build();

            SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ". Status
was "
                + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

Publikasikan pesan ke topik. Pesan dikirim ke setiap pelanggan.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

```

```
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <message> <phoneNumber>

            Where:
                message - The message text to send.
                phoneNumber - The mobile phone number to which a message is sent
(for example, +1XXX5550100).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .phoneNumber(phoneNumber)
                .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```



```
}
```

## Contoh nirserver

### Memanggil fungsi Lambda dari pemicu Amazon SNS

Contoh kode berikut menunjukkan cara menerapkan fungsi Lambda yang menerima peristiwa yang dipicu dengan menerima pesan dari topik SNS. Fungsi mengambil pesan dari parameter peristiwa dan mencatat konten setiap pesan.

### SDK for Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

### Mengkonsumsi acara SNS dengan Lambda menggunakan Java.

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;

import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
```

```
        Iterator<SNSRecord> recordsIter = records.iterator();
        while (recordsIter.hasNext()) {
            processRecord(recordsIter.next());
        }
    }
    return Boolean.TRUE;
}

public void processRecord(SNSRecord record) {
    try {
        String message = record.getSNS().getMessage();
        logger.log("message: " + message);
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
}
```

## Contoh Amazon SQS menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan Amazon SQS. AWS SDK for Java 2.x

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Memulai

## Halo Amazon SQS

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon SQS.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.SqsException;
import software.amazon.awssdk.services.sqs.paginators.ListQueuesIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloSQS {
    public static void main(String[] args) {
        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();

        listQueues(sqsClient);
        sqsClient.close();
    }

    public static void listQueues(SqsClient sqsClient) {
        try {
            ListQueuesIterable listQueues = sqsClient.listQueuesPaginator();
            listQueues.stream()
                .flatMap(r -> r.queueUrls().stream())
                .forEach(content -> System.out.println(" Queue URL: " +
                    content.toLowerCase()));
        }
    }
}
```

```
    } catch (SqsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}  
}
```

- Untuk detail API, lihat [ListQueues](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

### Membuat antrian

Contoh kode berikut menunjukkan cara membuat antrian Amazon SQS.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sqs.SqsClient;  
import software.amazon.awssdk.services.sqs.model.ChangeMessageVisibilityRequest;  
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;  
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;  
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;  
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;  
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;  
import software.amazon.awssdk.services.sqs.model.ListQueuesResponse;  
import software.amazon.awssdk.services.sqs.model.Message;  
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
```

```
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SQSExample {
    public static void main(String[] args) {
        String queueName = "queue" + System.currentTimeMillis();
        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();

        // Perform various tasks on the Amazon SQS queue.
        String queueUrl = createQueue(sqsClient, queueName);
        listQueues(sqsClient);
        listQueuesFilter(sqsClient, queueUrl);
        List<Message> messages = receiveMessages(sqsClient, queueUrl);
        sendBatchMessages(sqsClient, queueUrl);
        changeMessages(sqsClient, queueUrl, messages);
        deleteMessages(sqsClient, queueUrl, messages);
        sqsClient.close();
    }

    public static String createQueue(SqsClient sqsClient, String queueName) {
        try {
            System.out.println("\nCreate Queue");

            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                .queueName(queueName)
                .build();

            sqsClient.createQueue(createQueueRequest);

            System.out.println("\nGet queue url");
        }
    }
}
```

```
        GetQueueUrlResponse getQueueUrlResponse = sqsClient
        .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        return getQueueUrlResponse.queueUrl();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void listQueues(SqsClient sqsClient) {

    System.out.println("\nList Queues");
    String prefix = "que";

    try {
        ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
        ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
        for (String url : listQueuesResponse.queueUrls()) {
            System.out.println(url);
        }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void listQueuesFilter(SqsClient sqsClient, String queueUrl) {
    // List queues with filters
    String namePrefix = "queue";
    ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
        .queueNamePrefix(namePrefix)
        .build();

    ListQueuesResponse listQueuesFilteredResponse =
sqsClient.listQueues(filterListRequest);
    System.out.println("Queue URLs with prefix: " + namePrefix);
    for (String url : listQueuesFilteredResponse.queueUrls()) {
        System.out.println(url);
    }
}
```

```
    }

    System.out.println("\nSend message");
    try {
        sqsClient.sendMessage(SendMessageRequest.builder()
            .queueUrl(queueUrl)
            .messageBody("Hello world!")
            .delaySeconds(10)
            .build());
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

    System.out.println("\nSend multiple messages");
    try {
        SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
            .queueUrl(queueUrl)

            .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)

                .build())
            .build();
        sqsClient.sendMessageBatch(sendMessageBatchRequest);
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl) {

    System.out.println("\nReceive messages");
    try {
```

```
        ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
            .queueUrl(queueUrl)
            .numberOfMessages(5)
            .build();
        return sqsClient.receiveMessage(receiveMessageRequest).messages();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static void changeMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {

    System.out.println("\nChange Message Visibility");
    try {

        for (Message message : messages) {
            ChangeMessageVisibilityRequest req =
ChangeMessageVisibilityRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .visibilityTimeout(100)
                .build();
            sqsClient.changeMessageVisibility(req);
        }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
    System.out.println("\nDelete Messages");

    try {
        for (Message message : messages) {
            DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
```



```
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .build();
            sqsClient.deleteMessage(deleteMessageRequest);
        }
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [CreateQueue](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus pesan dari antrian

Contoh kode berikut menunjukkan cara menghapus pesan dari antrian Amazon SQS.

SDK for Java 2.x

### Note

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

```
try {
    for (Message message : messages) {
        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .build();
        sqsClient.deleteMessage(deleteMessageRequest);
    }
} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

- Untuk detail API, lihat [DeleteMessage](#) di Referensi AWS SDK for Java 2.x API.

## Hapus antrian

Contoh kode berikut menunjukkan cara menghapus antrian Amazon SQS.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteQueue {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <queueName>

                Where:
                    queueName - The name of the Amazon SQS queue to delete.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String queueName = args[0];
    SqsClient sqs = SqsClient.builder()
        .region(Region.US_WEST_2)
        .build();

    deleteSQSQueue(sqs, queueName);
    sqs.close();
}

public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
    try {
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);


    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DeleteQueue](#) di Referensi AWS SDK for Java 2.x API.

## Dapatkan URL antrian

Contoh kode berikut menunjukkan cara mendapatkan URL antrian Amazon SQS.

## SDK for Java 2.x

 Note

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


```
GetQueueUrlResponse getQueueUrlResponse = sqsClient
    .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
    return getQueueUrlResponse.queueUrl();
```

- Untuk detail API, lihat [GetQueueUrl](#) di Referensi AWS SDK for Java 2.x API.

## Daftar antrian

Contoh kode berikut menunjukkan cara membuat daftar antrian Amazon SQS.

## SDK for Java 2.x

 Note

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

```
String prefix = "que";

try {
    ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
    ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
    for (String url : listQueuesResponse.queueUrls()) {
        System.out.println(url);
    }
}
```

```
    } catch (SqsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Untuk detail API, lihat [ListQueues](#) di Referensi AWS SDK for Java 2.x API.

## Menerima pesan dari antrian

Contoh kode berikut menunjukkan cara menerima pesan dari antrian Amazon SQS.

### SDK for Java 2.x

#### Note

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


```
    try {  
        ReceiveMessageRequest receiveMessageRequest =  
ReceiveMessageRequest.builder()  
            .queueUrl(queueUrl)  
            .numberOfMessages(5)  
            .build();  
        return sqsClient.receiveMessage(receiveMessageRequest).messages();  
    } catch (SqsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return null;  
}
```

- Untuk detail API, lihat [ReceiveMessage](#) di Referensi AWS SDK for Java 2.x API.

## Mengirim sekumpulan pesan ke antrian

Contoh kode berikut menunjukkan cara mengirim sekumpulan pesan ke antrian Amazon SQS.

## SDK for Java 2.x

 Note

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

```
SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
    .queueUrl(queueUrl)

    .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)
        .build())
    .build();
sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

- Untuk detail API, lihat [SendMessageBatch](#) di Referensi AWS SDK for Java 2.x API.

## Mengirim Pesan Antrian

Contoh kode berikut menunjukkan cara mengirim pesan ke antrian Amazon SQS.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
```

```
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessages {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <queueName> <message>

            Where:
                queueName - The name of the queue.
                message - The message to send.
            "";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        String message = args[1];
        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();
        sendMessage(sqsClient, queueName, message);
        sqsClient.close();
    }

    public static void sendMessage(SqsClient sqsClient, String queueName, String
message) {
        try {
            CreateQueueRequest request = CreateQueueRequest.builder()
                .queueName(queueName)
                .build();
            sqsClient.createQueue(request);
        }
    }
}
```

```
GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
    .queueName(queueName)
    .build();

String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
SendMessageRequest sendMsgRequest = SendMessageRequest.builder()
    .queueUrl(queueUrl)
    .messageBody(message)
    .delaySeconds(5)
    .build();

sqsClient.sendMessage(sendMsgRequest);

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Untuk detail API, lihat [SendMessage](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

Buat dan publikasikan ke topik FIFO

Contoh kode berikut menunjukkan cara membuat dan mempublikasikan ke topik FIFO Amazon SNS.

SDK for Java 2.x

### Note

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

Contoh ini

- membuat topik Amazon SNS FIFO, dua antrian FIFO Amazon SQS, dan satu antrian Standar.
- berlangganan antrian ke topik dan menerbitkan pesan ke topik tersebut.



[Tes](#) memverifikasi penerimaan pesan ke setiap antrian. [Contoh lengkap](#) juga menunjukkan penambahan kebijakan akses dan menghapus sumber daya di akhir.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
            "Where:\n" +
            "    fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
            "    wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
            +
            "    retailQueueARN - The name of a SQS FIFO queue that will created
for the retail consumer. \n\n" +
            "    analyticsQueueARN - The name of a SQS standard queue that will
be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue: ARN,
URL,
        // name and type.
        List<QueueData> queues = List.of(
            new QueueData(wholeSaleQueueName, QueueType.FIFO),
            new QueueData(retailQueueName, QueueType.FIFO),
            new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);
    }
}
```

```
// Create a topic.
String topicARN = createFIFOTopic(fifoTopicName);

// Subscribe each queue to the topic.
subscribeQueues(queues, topicARN);

// Allow the newly created topic to send messages to the queues.
addAccessPolicyToQueuesFINAL(queues, topicARN);

// Publish a sample price update message with payload.
publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

// Clean up resources.
deleteSubscriptions(queues);
deleteQueues(queues);
deleteTopic(topicARN);
}

public static String createFIFOTopic(String topicName) {
    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = Map.of(
            "FifoTopic", "true",
            "ContentBasedDeduplication", "false");

        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
            .name(topicName)
            .attributes(topicAttributes)
            .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        String topicArn = response.topicArn();
        System.out.println("The topic ARN is" + topicArn);

        return topicArn;
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

```
public static void subscribeQueues(List<QueueData> queues, String topicARN) {
    queues.forEach(queue -> {
        SubscribeRequest subscribeRequest = SubscribeRequest.builder()
            .topicArn(topicARN)
            .endpoint(queue.queueARN)
            .protocol("sqs")
            .build();

        // Subscribe to the endpoint by using the SNS service client.
        // Only Amazon SQS queues can receive notifications from an Amazon SNS
        FIFO
        // topic.
        SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
        System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]);
        queue.subscriptionARN = subscribeResponse.subscriptionArn();
    });
}

public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

    try {
        // Create and publish a message that updates the wholesale price.
        String subject = "Price Update";
        String dedupId = UUID.randomUUID().toString();
        String attributeName = "business";
        String attributeValue = "wholesale";

        MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
            .dataType("String")
            .stringValue(attributeValue)
            .build();

        Map<String, MessageAttributeValue> attributes = new HashMap<>();
        attributes.put(attributeName, msgAttValue);
        PublishRequest pubRequest = PublishRequest.builder()
            .topicArn(topicArn)
            .subject(subject)
            .message(payload)
            .messageGroupId(groupId)
            .messageDeduplicationId(dedupId)
```

```
        .messageAttributes(attributes)
        .build();

    final PublishResponse response = snsClient.publish(pubRequest);
    System.out.println(response.messageId());
    System.out.println(response.sequenceNumber());
    System.out.println("Message was published to " + topicArn);

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [CreateTopic](#)
  - [Publikasikan](#)
  - [Berlangganan](#)

## Contoh nirserver

### Memanggil fungsi Lambda dari pemicu Amazon SQS

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menerima peristiwa yang dipicu oleh menerima pesan dari antrian SQS. Fungsi mengambil pesan dari parameter peristiwa dan mencatat konten setiap pesan.

### SDK for Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

### Mengonsumsi acara SQS dengan Lambda menggunakan Java.

```
import com.amazonaws.services.lambda.runtime.Context;
```

```
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }
    }
}
```

## Melaporkan kegagalan item batch untuk fungsi Lambda dengan pemicu Amazon SQS

Contoh kode berikut menunjukkan cara mengimplementasikan respons batch sebagian untuk fungsi Lambda yang menerima peristiwa dari antrian SQS. Fungsi melaporkan kegagalan item batch dalam respons, memberi sinyal ke Lambda untuk mencoba lagi pesan tersebut nanti.

### SDK for Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

## Melaporkan kegagalan item batch SQS dengan Lambda menggunakan Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {

        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
ArrayList<SQSBatchResponse.BatchItemFailure>();
        String messageId = "";
        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
                messageId = message.getMessageId();
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
SQSBatchResponse.BatchItemFailure(messageId));
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }
}
```

## Contoh Step Functions menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with Step Functions.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.


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

Memulai

Hello Step Functions

Contoh kode berikut menunjukkan cara memulai menggunakan Step Functions.

SDK for Java 2.x

 Note

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

Hello versi Java.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
```

```
        .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
                System.out.println("The name of the state machine is: " +
machine.name());
                System.out.println("The ARN value is : " +
machine.stateMachineArn());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat [ListStateMachines](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

### Buat mesin status

Contoh kode berikut menunjukkan cara membuat mesin status Step Functions.



## SDK for Java 2.x

 Note

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

```
public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
    try {
        CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
            .definition(json)
            .name(stateMachineName)
            .roleArn(roleARN)
            .type(StateMachineType.STANDARD)
            .build();

        CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
        return response.stateMachineArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Untuk detail API, lihat [CreateStateMachine](#) di Referensi AWS SDK for Java 2.x API.

## Buat aktivitas

Contoh kode berikut menunjukkan cara membuat aktivitas Step Functions.

## SDK for Java 2.x

 Note

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

```
public static String createActivity(SfnClient sfnClient, String activityName) {
    try {
        CreateActivityRequest activityRequest = CreateActivityRequest.builder()
            .name(activityName)
            .build();

        CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
        return response.activityArn();


    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Untuk detail API, lihat [CreateActivity](#) di Referensi AWS SDK for Java 2.x API.

## Hapus mesin negara

Contoh kode berikut menunjukkan cara menghapus mesin status Step Functions.

## SDK for Java 2.x

 Note

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

```
public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
    try {
        DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
        .stateMachineArn(stateMachineArn)
        .build();

        sfnClient.deleteStateMachine(deleteStateMachineRequest);
        DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
        .stateMachineArn(stateMachineArn)
        .build();

        while (true) {
            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
            System.out.println("The state machine is not deleted yet. The status
is " + response.status());
            Thread.sleep(3000);
        }

    } catch (SfnException | InterruptedException e) {
        System.err.println(e.getMessage());
    }
    System.out.println(stateMachineArn + " was successfully deleted.");
}
```

- Untuk detail API, lihat [DeleteStateMachine](#) di Referensi AWS SDK for Java 2.x API.

## Menghapus aktivitas

Contoh kode berikut menunjukkan cara menghapus aktivitas Step Functions.

### SDK for Java 2.x

#### Note

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

```
public static void deleteActivity(SfnClient sfnClient, String actArn) {
    try {
        DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
            .activityArn(actArn)
            .build();

        sfnClient.deleteActivity(activityRequest);
        System.out.println("You have deleted " + actArn);


    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DeleteActivity](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan mesin negara

Contoh kode berikut menunjukkan bagaimana mendeskripsikan mesin status Step Functions.

SDK for Java 2.x

 Note

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

```
public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
    try {
        DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
```

```
        System.out.println("The name of the State machine is " +
response.name());
        System.out.println("The status of the State machine is " +
response.status());
        System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
        System.out.println("The role ARN value is " + response.roleArn());


    } catch (SfnException e) {
        System.err.println(e.getMessage());
    }
}
```

- Untuk detail API, lihat [DescribeStateMachine](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan mesin negara yang dijalankan

Contoh kode berikut menunjukkan bagaimana mendeskripsikan Step Functions state machine run.

SDK for Java 2.x

 Note

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

```
public static void describeExe(SfnClient sfnClient, String executionArn) {
    try {
        DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
            .executionArn(executionArn)
            .build();

        String status = "";
        boolean hasSucceeded = false;
        while (!hasSucceeded) {
            DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
            status = response.statusAsString();
        }
    }
}
```

```

        if (status.compareTo("RUNNING") == 0) {
            System.out.println("The state machine is still running, let's
wait for it to finish.");
            Thread.sleep(2000);
        } else if (status.compareTo("SUCCEEDED") == 0) {
            System.out.println("The Step Function workflow has succeeded");
            hasSucceeded = true;
        } else {
            System.out.println("The Status is neither running or
succeeded");
        }
    }
    System.out.println("The Status is " + status);

} catch (SfnException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
}

```

- Untuk detail API, lihat [DescribeExecution](#) di Referensi AWS SDK for Java 2.x API.

## Mendapatkan data tugas untuk suatu aktivitas

Contoh kode berikut menunjukkan cara mendapatkan data tugas untuk aktivitas Step Functions.

### SDK for Java 2.x

#### Note

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

```

public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
    List<String> myList = new ArrayList<>();
    GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
        .activityArn(actArn)
        .build();
}

```

```
        GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
        myList.add(response.taskToken());
        myList.add(response.input());
        return myList;
    }

    /// <summary>
    /// Stop execution of a Step Functions workflow.
    /// </summary>
    /// <param name="executionArn">The Amazon Resource Name (ARN) of
    /// the Step Functions execution to stop.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> StopExecution(string executionArn)
    {
        var response =
            await _amazonStepFunctions.StopExecutionAsync(new StopExecutionRequest
            { ExecutionArn = executionArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}
```

- Untuk detail API, lihat [GetActivityTask](#) di Referensi AWS SDK for Java 2.x API.

## Daftar kegiatan

Contoh kode berikut menunjukkan cara membuat daftar aktivitas Step Functions.

## SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListActivitiesRequest;
import software.amazon.awssdk.services.sfn.model.ListActivitiesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
```

```
import software.amazon.awssdk.services.sfn.model.ActivityListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListActivities {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        listAllActivites(sfnClient);
        sfnClient.close();
    }

    public static void listAllActivites(SfnClient sfnClient) {
        try {
            ListActivitiesRequest activitiesRequest =
ListActivitiesRequest.builder()
                .maxResults(10)
                .build();

            ListActivitiesResponse response =
sfnClient.listActivities(activitiesRequest);
            List<ActivityListItem> items = response.activities();
            for (ActivityListItem item : items) {
                System.out.println("The activity ARN is " + item.activityArn());
                System.out.println("The activity name is " + item.name());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```



- Untuk detail API, lihat [ListActivities](#) di Referensi AWS SDK for Java 2.x API.

## Daftar mesin status berjalan

Contoh kode berikut menunjukkan bagaimana daftar Step Functions state machine berjalan.

## SDK for Java 2.x

### Note

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

```
public static void getExeHistory(SfnClient sfnClient, String exeARN) {
    try {
        GetExecutionHistoryRequest historyRequest =
        GetExecutionHistoryRequest.builder()
            .executionArn(exeARN)
            .maxResults(10)
            .build();

        GetExecutionHistoryResponse historyResponse =
        sfnClient.getExecutionHistory(historyRequest);
        List<HistoryEvent> events = historyResponse.events();
        for (HistoryEvent event : events) {
            System.out.println("The event type is " + event.type().toString());
        }

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ListExecutions](#) di Referensi AWS SDK for Java 2.x API.

## Daftar mesin negara

Contoh kode berikut menunjukkan cara mencantumkan mesin status Step Functions.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
```

```
        System.out.println("The name of the state machine is: " +
machine.name());
        System.out.println("The ARN value is : " +
machine.stateMachineArn());
    }


    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [ListStateMachines](#) di Referensi AWS SDK for Java 2.x API.

Kirim respons sukses ke suatu tugas

Contoh kode berikut menunjukkan cara mengirim respons sukses ke tugas Step Functions.

SDK for Java 2.x

 Note

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

```
public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
    try {
        SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
            .taskToken(token)
            .output(json)
            .build();

        sfnClient.sendTaskSuccess(successRequest);

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```


```
}
```

- Untuk detail API, lihat [SendTaskSuccess](#) di Referensi AWS SDK for Java 2.x API.

Mulai menjalankan mesin status

Contoh kode berikut menunjukkan bagaimana memulai Step Functions state machine run.

SDK for Java 2.x

 Note

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

```
public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
    UUID uuid = UUID.randomUUID();
    String uuidValue = uuid.toString();
    try {
        StartExecutionRequest executionRequest = StartExecutionRequest.builder()
            .input(jsonEx)
            .stateMachineArn(stateMachineArn)
            .name(uuidValue)
            .build();

        StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
        return response.executionArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Untuk detail API, lihat [StartExecution](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

Memulai dengan mesin negara

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat aktivitas.
- Buat mesin status dari definisi Amazon States Language 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 for Java 2.x

### Note

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

```
/**
 * You can obtain the JSON file to create a state machine in the following
 * GitHub location.
 *
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files
 *
 * To run this code example, place the chat_sfn_state_machine.json file into
 * your project's resources folder.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an activity.
 * 2. Creates a state machine.
 * 3. Describes the state machine.
```

```

* 4. Starts execution of the state machine and interacts with it.
* 5. Describes the execution.
* 6. Delete the activity.
* 7. Deletes the state machine.
*/
public class StepFunctionsScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws Exception {
        final String usage = ""

            Usage:
                <roleARN> <activityName> <stateMachineName>

            Where:
                roleName - The name of the IAM role to create for this state
machine.
                activityName - The name of an activity to create.
                stateMachineName - The name of the state machine to create.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String activityName = args[1];
        String stateMachineName = args[2];
        String polJSON = "{\n" +
            "    \"Version\": \"2012-10-17\",\n" +
            "    \"Statement\": [\n" +
            "        {\n" +
            "            \"Sid\": \"\",\n" +
            "            \"Effect\": \"Allow\",\n" +
            "            \"Principal\": {\n" +
            "                \"Service\": \"states.amazonaws.com\"\n" +
            "            },\n" +
            "            \"Action\": \"sts:AssumeRole\"\n" +
            "        }\n" +
            "    ]\n" +
            "    }";

        Scanner sc = new Scanner(System.in);

```

```
boolean action = false;

Region region = Region.US_EAST_1;
SfnClient sfncClient = SfnClient.builder()
    .region(region)
    .build();

Region regionGl = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(regionGl)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the AWS Step Functions example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an activity.");
String activityArn = createActivity(sfncClient, activityName);
System.out.println("The ARN of the activity is " + activityArn);
System.out.println(DASHES);

// Get JSON to use for the state machine and place the activityArn value
into
// it.
InputStream input = StepFunctionsScenario.class.getClassLoader()
    .getResourceAsStream("chat_sfn_state_machine.json");
ObjectMapper mapper = new ObjectMapper();
JsonNode jsonNode = mapper.readValue(input, JsonNode.class);
String jsonString = mapper.writeValueAsString(jsonNode);

// Modify the Resource node.
ObjectMapper objectMapper = new ObjectMapper();
JsonNode root = objectMapper.readTree(jsonString);
((ObjectNode) root.path("States").path("GetInput")).put("Resource",
activityArn);

// Convert the modified Java object back to a JSON string.
String stateDefinition = objectMapper.writeValueAsString(root);
System.out.println(stateDefinition);

System.out.println(DASHES);
System.out.println("2. Create a state machine.");
String roleARN = createIAMRole(iam, roleName, polJSON);
```

```
String stateMachineArn = createMachine(sfnClient, roleARN, stateMachineName,
stateDefinition);
System.out.println("The ARN of the state machine is " + stateMachineArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Describe the state machine.");
describeStateMachine(sfnClient, stateMachineArn);
System.out.println("What should ChatSFN call you?");
String userName = sc.nextLine();
System.out.println("Hello " + userName);
System.out.println(DASHES);

System.out.println(DASHES);
// The JSON to pass to the StartExecution call.
String executionJson = "{ \"name\" : \"" + userName + "\" }";
System.out.println(executionJson);
System.out.println("4. Start execution of the state machine and interact
with it.");
String runArn = startWorkflow(sfnClient, stateMachineArn, executionJson);
System.out.println("The ARN of the state machine execution is " + runArn);
List<String> myList;
while (!action) {
    myList = getActivityTask(sfnClient, activityArn);
    System.out.println("ChatSFN: " + myList.get(1));
    System.out.println(userName + " please specify a value.");
    String myAction = sc.nextLine();
    if (myAction.compareTo("done") == 0)
        action = true;

    System.out.println("You have selected " + myAction);
    String taskJson = "{ \"action\" : \"" + myAction + "\" }";
    System.out.println(taskJson);
    sendTaskSuccess(sfnClient, myList.get(0), taskJson);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Describe the execution.");
describeExe(sfnClient, runArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Delete the activity.");
```



```
deleteActivity(sfnClient, activityArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the state machines.");
deleteMachine(sfnClient, stateMachineArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The AWS Step Functions example scenario is complete.");
System.out.println(DASHES);
}

public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(polJSON)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        return response.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void describeExe(SfnClient sfnClient, String executionArn) {
    try {
        DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
            .executionArn(executionArn)
            .build();

        String status = "";
        boolean hasSucceeded = false;
        while (!hasSucceeded) {
            DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
```

```
        status = response.statusAsString();
        if (status.compareTo("RUNNING") == 0) {
            System.out.println("The state machine is still running, let's
wait for it to finish.");
            Thread.sleep(2000);
        } else if (status.compareTo("SUCCEEDED") == 0) {
            System.out.println("The Step Function workflow has succeeded");
            hasSucceeded = true;
        } else {
            System.out.println("The Status is neither running or
succeeded");
        }
    }
    System.out.println("The Status is " + status);

} catch (SfnException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

}

public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
    try {
        SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
            .taskToken(token)
            .output(json)
            .build();

        sfnClient.sendTaskSuccess(successRequest);

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
    List<String> myList = new ArrayList<>();
    GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
        .activityArn(actArn)
        .build();
```

```
        GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
        myList.add(response.taskToken());
        myList.add(response.input());
        return myList;
    }

    public static void deleteActivity(SfnClient sfnClient, String actArn) {
        try {
            DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
                .activityArn(actArn)
                .build();

            sfnClient.deleteActivity(activityRequest);
            System.out.println("You have deleted " + actArn);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
        try {
            DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
            System.out.println("The name of the State machine is " +
response.name());
            System.out.println("The status of the State machine is " +
response.status());
            System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
            System.out.println("The role ARN value is " + response.roleArn());

        } catch (SfnException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

```
public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
    try {
        DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        sfnClient.deleteStateMachine(deleteStateMachineRequest);
        DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        while (true) {
            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
            System.out.println("The state machine is not deleted yet. The status
is " + response.status());
            Thread.sleep(3000);
        }

    } catch (SfnException | InterruptedException e) {
        System.err.println(e.getMessage());
    }
    System.out.println(stateMachineArn + " was successfully deleted.");
}

public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
    UUID uuid = UUID.randomUUID();
    String uuidValue = uuid.toString();
    try {
        StartExecutionRequest executionRequest = StartExecutionRequest.builder()
            .input(jsonEx)
            .stateMachineArn(stateMachineArn)
            .name(uuidValue)
            .build();

        StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
        return response.executionArn();

    } catch (SfnException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
    try {
        CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
            .definition(json)
            .name(stateMachineName)
            .roleArn(roleARN)
            .type(StateMachineType.STANDARD)
            .build();

        CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
        return response.stateMachineArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createActivity(SfnClient sfnClient, String activityName) {
    try {
        CreateActivityRequest activityRequest = CreateActivityRequest.builder()
            .name(activityName)
            .build();

        CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
        return response.activityArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

```
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [CreateActivity](#)
  - [CreateStateMachine](#)
  - [DeleteActivity](#)
  - [DeleteStateMachine](#)
  - [DescribeExecution](#)
  - [DescribeStateMachine](#)
  - [GetActivityTask](#)
  - [ListActivities](#)
  - [ListStateMachines](#)
  - [SendTaskSuccess](#)
  - [StartExecution](#)
  - [StopExecution](#)

## AWS STS contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with AWS STS.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

## Tindakan

Asumsikan peran

Contoh kode berikut menunjukkan bagaimana untuk mengambil peran dengan AWS STS.

SDK for Java 2.x

### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 *   "Version": "2012-10-17",
 *   "Statement": [
 *     {
 *       "Effect": "Allow",
 *       "Principal": {
 *         "AWS": "<Specify the ARN of your IAM user you are using in this code
 * example>"
 *       },
 *       "Action": "sts:AssumeRole"
 *     }
 *   ]
 * }
```

```

* }
*
* For more information, see "Editing the Trust Relationship for an Existing
* Role" in the AWS Directory Service guide.
*
* Also, set up your development environment, including your credentials.
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <roleArn> <roleSessionName>\s

            Where:
                roleArn - The Amazon Resource Name (ARN) of the role to assume
(for example, rn:aws:iam::000008047983:role/s3role).\s
                roleSessionName - An identifier for the assumed role session
(for example, mysession).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
        StsClient stsClient = StsClient.builder()
            .region(region)
            .build();

        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }

    public static void assumeGivenRole(StsClient stsClient, String roleArn, String
roleSessionName) {
        try {

```



```
AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
    .roleArn(roleArn)
    .roleSessionName(roleSessionName)
    .build();

AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
Credentials myCreds = roleResponse.credentials();

// Display the time when the temp creds expire.
Instant exTime = myCreds.expiration();
String tokenInfo = myCreds.sessionToken();

// Convert the Instant to readable date.
DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
    .withLocale(Locale.US)
    .withZone(ZoneId.systemDefault());

formatter.format(exTime);
System.out.println("The token " + tokenInfo + " expires on " + exTime);

} catch (StsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Untuk detail API, lihat [AssumeRole](#) di Referensi AWS SDK for Java 2.x API.

## AWS Support contoh menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with AWS Support.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.


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

Memulai

Halo AWS Support

Contoh kode berikut menunjukkan cara untuk mulai menggunakan AWS Support.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SupportException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following task:
 *
 * 1. Gets and displays available services.
```

```
*
*
* NOTE: To see multiple operations, see SupportScenario.
*/

public class HelloSupport {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
            .region(region)
            .build();

        System.out.println("***** Step 1. Get and display available services.");
        displayServices(supportClient);
    }

    // Return a List that contains a Service name and Category name.
    public static void displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
                .language("en")
                .build();

            DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());

                // Display the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                }
                index++;
            }
        }
    }
}
```

```
        } catch (SupportException e) {  
            System.out.println(e.getLocalizedMessage());  
            System.exit(1);  
        }  
    }  
}
```

- Untuk detail API, lihat [DescribeServices](#) di Referensi AWS SDK for Java 2.x API.

## Topik

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

## Tindakan

Tambahkan komunikasi ke kasus

Contoh kode berikut menunjukkan cara menambahkan AWS Support komunikasi dengan lampiran ke kasus dukungan.

SDK for Java 2.x

### Note

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

```
public static void addAttachSupportCase(SupportClient supportClient, String  
caseId, String attachmentSetId) {  
    try {  
        AddCommunicationToCaseRequest caseRequest =  
AddCommunicationToCaseRequest.builder()  
            .caseId(caseId)  
            .attachmentSetId(attachmentSetId)  
            .communicationBody("Please refer to attachment for details.")  
            .build();
```

```
        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
        if (response.result())
            System.out.println("You have successfully added a communication to
an AWS Support case");
        else
            System.out.println("There was an error adding the communication to
an AWS Support case");


    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [AddCommunicationToCase](#) di Referensi AWS SDK for Java 2.x API.

Tambahkan lampiran ke satu set

Contoh kode berikut menunjukkan cara menambahkan AWS Support lampiran ke set lampiran.

SDK for Java 2.x

 Note

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

```
public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
    try {
        File myFile = new File(fileAttachment);
        InputStream sourceStream = new FileInputStream(myFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        Attachment attachment = Attachment.builder()
            .fileName(myFile.getName())
            .data(sourceBytes)
            .build();
```

```

        AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
        .attachments(attachment)
        .build();

        AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
        return response.attachmentSetId();

    } catch (SupportException | FileNotFoundException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

```

- Untuk detail API, lihat [AddAttachmentsToSet](#) di Referensi AWS SDK for Java 2.x API.

## Buat kasus

Contoh kode berikut menunjukkan cara membuat AWS Support kasus baru.

## SDK for Java 2.x

### Note

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

```

public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
    try {
        String serviceCode = sevCatList.get(0);
        String caseCat = sevCatList.get(1);
        CreateCaseRequest caseRequest = CreateCaseRequest.builder()
            .categoryCode(caseCat.toLowerCase())
            .serviceCode(serviceCode.toLowerCase())
            .severityCode(sevLevel.toLowerCase())
            .communicationBody("Test issue with " +
serviceCode.toLowerCase())

```

```
        .subject("Test case, please ignore")
        .language("en")
        .issueType("technical")
        .build();

        CreateCaseResponse response = supportClient.createCase(caseRequest);
        return response.caseId();

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- Untuk detail API, lihat [CreateCase](#) di Referensi AWS SDK for Java 2.x API.

## Jelaskan lampiran

Contoh kode berikut menunjukkan bagaimana mendeskripsikan lampiran untuk sebuah AWS Support kasus.

## SDK for Java 2.x

### Note

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

```
public static void describeAttachment(SupportClient supportClient, String
attachId) {
    try {
        DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
            .attachmentId(attachId)
            .build();

        DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
```

```
        System.out.println("The name of the file is " +
response.attachment().fileName());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DescribeAttachment](#) di Referensi AWS SDK for Java 2.x API.

## Jelaskan kasus

Contoh kode berikut menunjukkan bagaimana menggambarkan AWS Support kasus.

## SDK for Java 2.x

### Note

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

```
public static void getOpenCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
            .maxResults(20)
            .afterTime(yesterday.toString())
            .beforeTime(now.toString())
            .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
```



```
        System.out.println("The case status is " + sinCase.status());
        System.out.println("The case Id is " + sinCase.caseId());
        System.out.println("The case subject is " + sinCase.subject());
    }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedName());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [DescribeCases](#) di Referensi AWS SDK for Java 2.x API.

## Jelaskan komunikasi

Contoh kode berikut menunjukkan bagaimana mendeskripsikan AWS Support komunikasi untuk suatu kasus.

## SDK for Java 2.x

### Note

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

```
public static String listCommunications(SupportClient supportClient, String
caseId) {
    try {
        String attachId = null;
        DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
            .caseId(caseId)
            .maxResults(10)
            .build();

        DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
        List<Communication> communications = response.communications();
        for (Communication comm : communications) {
```

```
        System.out.println("the body is: " + comm.body());

        // Get the attachment id value.
        List<AttachmentDetails> attachments = comm.attachmentSet();
        for (AttachmentDetails detail : attachments) {
            attachId = detail.attachmentId();
        }
    }
    return attachId;


} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return "";
}
```

- Untuk detail API, lihat [DescribeCommunications](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan layanan

Contoh kode berikut menunjukkan bagaimana menggambarkan daftar AWS layanan.

SDK for Java 2.x

 Note

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

```
// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
            .language("en")
            .build();

        DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
```

```
String serviceCode = null;
String catName = null;
List<String> sevCatList = new ArrayList<>();
List<Service> services = response.services();

System.out.println("Get the first 10 services");
int index = 1;
for (Service service : services) {
    if (index == 11)
        break;

    System.out.println("The Service name is: " + service.name());
    if (service.name().compareTo("Account") == 0)
        serviceCode = service.code();

    // Get the Categories for this service.
    List<Category> categories = service.categories();
    for (Category cat : categories) {
        System.out.println("The category name is: " + cat.name());
        if (cat.name().compareTo("Security") == 0)
            catName = cat.name();
    }
    index++;
}

// Push the two values to the list.
sevCatList.add(serviceCode);
sevCatList.add(catName);
return sevCatList;

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return null;
}
```

- Untuk detail API, lihat [DescribeServices](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan tingkat keparahan

Contoh kode berikut menunjukkan bagaimana menggambarkan AWS Support tingkat keparahan.

## SDK for Java 2.x

 Note

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

```
public static String displaySevLevels(SupportClient supportClient) {
    try {
        DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
            .language("en")
            .build();

        DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
        List<SeverityLevel> severityLevels = response.severityLevels();
        String levelName = null;
        for (SeverityLevel sevLevel : severityLevels) {
            System.out.println("The severity level name is: " +
sevLevel.name());
            if (sevLevel.name().compareTo("High") == 0)
                levelName = sevLevel.name();
        }
        return levelName;

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- Untuk detail API, lihat [DescribeSeverityLevels](#) di Referensi AWS SDK for Java 2.x API.

## Selesaikan kasus

Contoh kode berikut menunjukkan cara menyelesaikan AWS Support kasus.

## SDK for Java 2.x

 Note

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

```
public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
            .caseId(caseId)
            .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
        System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- Untuk detail API, lihat [ResolveCase](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

### Memulai kasus

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- 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 for Java 2.x

### Note

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

Jalankan berbagai AWS Support operasi.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetResponse;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseRequest;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseResponse;
import software.amazon.awssdk.services.support.model.Attachment;
import software.amazon.awssdk.services.support.model.AttachmentDetails;
import software.amazon.awssdk.services.support.model.CaseDetails;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.Communication;
import software.amazon.awssdk.services.support.model.CreateCaseRequest;
import software.amazon.awssdk.services.support.model.CreateCaseResponse;
import software.amazon.awssdk.services.support.model.DescribeAttachmentRequest;
import software.amazon.awssdk.services.support.model.DescribeAttachmentResponse;
import software.amazon.awssdk.services.support.model.DescribeCasesRequest;
import software.amazon.awssdk.services.support.model.DescribeCasesResponse;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsRequest;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsResponse;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsRequest;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsResponse;
import software.amazon.awssdk.services.support.model.ResolveCaseRequest;
import software.amazon.awssdk.services.support.model.ResolveCaseResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SeverityLevel;
import software.amazon.awssdk.services.support.model.SupportException;
```

```
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetRequest;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.time.Instant;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets and displays available services.
 * 2. Gets and displays severity levels.
 * 3. Creates a support case by using the selected service, category, and
 * severity level.
 * 4. Gets a list of open cases for the current day.
 * 5. Creates an attachment set with a generated file.
 * 6. Adds a communication with the attachment to the support case.
 * 7. Lists the communications of the support case.
 * 8. Describes the attachment set included with the communication.
 * 9. Resolves the support case.
 * 10. Gets a list of resolved cases for the current day.
 */
public class SupportScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = ""
```

```
Usage:
  <fileAttachment>Where:
  fileAttachment - The file can be a simple saved .txt file to use
as an email attachment.\s
  """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String fileAttachment = args[0];
Region region = Region.US_WEST_2;
SupportClient supportClient = SupportClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("***** Welcome to the AWS Support case example
scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Get and display available services.");
List<String> sevCatList = displayServices(supportClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Get and display Support severity levels.");
String sevLevel = displaySevLevels(supportClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Create a support case using the selected service,
category, and severity level.");
String caseId = createSupportCase(supportClient, sevCatList, sevLevel);
if (caseId.compareTo("") == 0) {
    System.out.println("A support case was not successfully created!");
    System.exit(1);
} else
    System.out.println("Support case " + caseId + " was successfully
created!");
System.out.println(DASHES);
```



```
        System.out.println(DASHES);
        System.out.println("4. Get open support cases.");
        getOpenCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create an attachment set with a generated file to add
to the case.");
        String attachmentSetId = addAttachment(supportClient, fileAttachment);
        System.out.println("The Attachment Set id value is" + attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add communication with the attachment to the support
case.");
        addAttachSupportCase(supportClient, caseId, attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. List the communications of the support case.");
        String attachId = listCommunications(supportClient, caseId);
        System.out.println("The Attachment id value is" + attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Describe the attachment set included with the
communication.");
        describeAttachment(supportClient, attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Resolve the support case.");
        resolveSupportCase(supportClient, caseId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of resolved cases for the current day.");
        getResolvedCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("***** This Scenario has successfully completed");
        System.out.println(DASHES);
    }
```

```
public static void getResolvedCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
            .maxResults(30)
            .afterTime(yesterday.toString())
            .beforeTime(now.toString())
            .includeResolvedCases(true)
            .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
            if (sinCase.status().compareTo("resolved") == 0)
                System.out.println("The case status is " + sinCase.status());
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
            .caseId(caseId)
            .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
        System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
  
  public static void describeAttachment(SupportClient supportClient, String  
attachId) {  
    try {  
      DescribeAttachmentRequest attachmentRequest =  
DescribeAttachmentRequest.builder()  
        .attachmentId(attachId)  
        .build();  
  
      DescribeAttachmentResponse response =  
supportClient.describeAttachment(attachmentRequest);  
      System.out.println("The name of the file is " +  
response.attachment().fileName());  
  
    } catch (SupportException e) {  
      System.out.println(e.getLocalizedMessage());  
      System.exit(1);  
    }  
  }  
  
  public static String listCommunications(SupportClient supportClient, String  
caseId) {  
    try {  
      String attachId = null;  
      DescribeCommunicationsRequest communicationsRequest =  
DescribeCommunicationsRequest.builder()  
        .caseId(caseId)  
        .maxResults(10)  
        .build();  
  
      DescribeCommunicationsResponse response =  
supportClient.describeCommunications(communicationsRequest);  
      List<Communication> communications = response.communications();  
      for (Communication comm : communications) {  
        System.out.println("the body is: " + comm.body());  
  
        // Get the attachment id value.  
        List<AttachmentDetails> attachments = comm.attachmentSet();  
        for (AttachmentDetails detail : attachments) {  
          attachId = detail.attachmentId();  
        }  
      }  
    }  
  }  
}
```

```
        return attachId;

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
    try {
        AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
            .caseId(caseId)
            .attachmentSetId(attachmentSetId)
            .communicationBody("Please refer to attachment for details.")
            .build();

        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
        if (response.result())
            System.out.println("You have successfully added a communication to
an AWS Support case");
        else
            System.out.println("There was an error adding the communication to
an AWS Support case");

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
    try {
        File myFile = new File(fileAttachment);
        InputStream sourceStream = new FileInputStream(myFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        Attachment attachment = Attachment.builder()
            .fileName(myFile.getName())
            .data(sourceBytes)
```

```
        .build();

        AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
        .attachments(attachment)
        .build();

        AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
        return response.attachmentSetId();

    } catch (SupportException | FileNotFoundException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static void getOpenCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
        .maxResults(20)
        .afterTime(yesterday.toString())
        .beforeTime(now.toString())
        .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
            System.out.println("The case status is " + sinCase.status());
            System.out.println("The case Id is " + sinCase.caseId());
            System.out.println("The case subject is " + sinCase.subject());
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
  
  public static String createSupportCase(SupportClient supportClient, List<String>  
sevCatList, String sevLevel) {  
    try {  
      String serviceCode = sevCatList.get(0);  
      String caseCat = sevCatList.get(1);  
      CreateCaseRequest caseRequest = CreateCaseRequest.builder()  
        .categoryCode(caseCat.toLowerCase())  
        .serviceCode(serviceCode.toLowerCase())  
        .severityCode(sevLevel.toLowerCase())  
        .communicationBody("Test issue with " +  
serviceCode.toLowerCase())  
        .subject("Test case, please ignore")  
        .language("en")  
        .issueType("technical")  
        .build();  
  
      CreateCaseResponse response = supportClient.createCase(caseRequest);  
      return response.caseId();  
  
    } catch (SupportException e) {  
      System.out.println(e.getLocalizedMessage());  
      System.exit(1);  
    }  
    return "";  
  }  
  
  public static String displaySevLevels(SupportClient supportClient) {  
    try {  
      DescribeSeverityLevelsRequest severityLevelsRequest =  
DescribeSeverityLevelsRequest.builder()  
        .language("en")  
        .build();  
  
      DescribeSeverityLevelsResponse response =  
supportClient.describeSeverityLevels(severityLevelsRequest);  
      List<SeverityLevel> severityLevels = response.severityLevels();  
      String levelName = null;  
      for (SeverityLevel sevLevel : severityLevels) {  
        System.out.println("The severity level name is: " +  
sevLevel.name());  
        if (sevLevel.name().compareTo("High") == 0)
```

```
        levelName = sevLevel.name();
    }
    return levelName;

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return "";
}

// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
            .language("en")
            .build();

        DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
        String serviceCode = null;
        String catName = null;
        List<String> sevCatList = new ArrayList<>();
        List<Service> services = response.services();

        System.out.println("Get the first 10 services");
        int index = 1;
        for (Service service : services) {
            if (index == 11)
                break;

            System.out.println("The Service name is: " + service.name());
            if (service.name().compareTo("Account") == 0)
                serviceCode = service.code();

            // Get the Categories for this service.
            List<Category> categories = service.categories();
            for (Category cat : categories) {
                System.out.println("The category name is: " + cat.name());
                if (cat.name().compareTo("Security") == 0)
                    catName = cat.name();
            }
            index++;
        }
    }
}
```

```
    }

    // Push the two values to the list.
    sevCatList.add(serviceCode);
    sevCatList.add(catName);
    return sevCatList;

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return null;
}
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [AddAttachmentsToSet](#)
  - [AddCommunicationToCase](#)
  - [CreateCase](#)
  - [DescribeAttachment](#)
  - [DescribeCases](#)
  - [DescribeCommunications](#)
  - [DescribeServices](#)
  - [DescribeSeverityLevels](#)
  - [ResolveCase](#)

## Contoh Systems Manager menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Java 2.x with Systems Manager.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.



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

## Topik

- [Tindakan](#)

## Tindakan

### Tambahkan parameter

Contoh kode berikut menunjukkan cara menambahkan parameter Systems Manager.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.ParameterType;
import software.amazon.awssdk.services.ssm.model.PutParameterRequest;
import software.amazon.awssdk.services.ssm.model.SsmException;

public class PutParameter {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <paraName>

            Where:
                paraName - The name of the parameter.
                paraValue - The value of the parameter.
            """;

        if (args.length != 2) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String paraName = args[0];
    String paraValue = args[1];
    Region region = Region.US_EAST_1;
    SsmClient ssmClient = SsmClient.builder()
        .region(region)
        .build();

    putParaValue(ssmClient, paraName, paraValue);
    ssmClient.close();
}

public static void putParaValue(SsmClient ssmClient, String paraName, String
value) {
    try {
        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(paraName)
            .type(ParameterType.STRING)
            .value(value)
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.println("The parameter was successfully added.");

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [PutParameter](#) di Referensi AWS SDK for Java 2.x API.

## Buat yang baru OpsItem

Contoh kode berikut menunjukkan cara membuat yang baru OpsItem.

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.CreateOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.CreateOpsItemResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateOpsItem {
    public static void main(String[] args) {

        final String USAGE = ""

            Usage:
                <title> <source> <category> <severity>

            Where:
                title - The OpsItem title.
                source - The origin of the OpsItem, such as Amazon EC2 or AWS
Systems Manager.
                category - A category to assign to an OpsItem.
                severity - A severity to assign to an OpsItem.

            """;

        if (args.length != 4) {
            System.out.println(USAGE);
            System.exit(1);
        }
    }
}
```

```
String title = args[0];
String source = args[1];
String category = args[2];
String severity = args[3];

Region region = Region.US_EAST_1;
SsmClient ssmClient = SsmClient.builder()
    .region(region)
    .build();

System.out
    .println("The Id of the OpsItem is " + createNewOpsItem(ssmClient,
title, source, category, severity));
    ssmClient.close();
}

public static String createNewOpsItem(SsmClient ssmClient,
    String title,
    String source,
    String category,
    String severity) {

    try {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
            .description("Created by the SSM Java API")
            .title(title)
            .source(source)
            .category(category)
            .severity(severity)
            .build();

        CreateOpsItemResponse itemResponse =
ssmClient.createOpsItem(opsItemRequest);
        return itemResponse.opsItemId();


    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Untuk detail API, lihat [CreateOpsItem](#) di Referensi AWS SDK for Java 2.x API.

Jelaskan sebuah OpsItem

Contoh kode berikut menunjukkan bagaimana mendeskripsikan sebuah OpsItem.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.DescribeOpsItemsRequest;
import software.amazon.awssdk.services.ssm.model.DescribeOpsItemsResponse;
import software.amazon.awssdk.services.ssm.model.OpsItemSummary;
import software.amazon.awssdk.services.ssm.model.SsmException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeOpsItems {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        describeItems(ssmClient);
        ssmClient.close();
    }
}
```

```
}

public static void describeItems(SsmClient ssmClient) {
    try {
        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
            .maxResults(10)
            .build();

        DescribeOpsItemsResponse itemsResponse =
            ssmClient.describeOpsItems(itemsRequest);
        List<OpsItemSummary> items = itemsResponse.opsItemSummaries();
        for (OpsItemSummary item : items) {
            System.out.println("The item title is " + item.title());
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DescribeOpsItems](#) di Referensi AWS SDK for Java 2.x API.

Dapatkan informasi parameter

Contoh kode berikut menunjukkan cara mendapatkan informasi parameter Systems Manager.

SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.GetParameterRequest;
import software.amazon.awssdk.services.ssm.model.GetParameterResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetParameter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <paraName>

            Where:
                paraName - The name of the parameter.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String paraName = args[0];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        getParaValue(ssmClient, paraName);
        ssmClient.close();
    }

    public static void getParaValue(SsmClient ssmClient, String paraName) {
        try {
            GetParameterRequest parameterRequest = GetParameterRequest.builder()
                .name(paraName)
                .build();

            GetParameterResponse parameterResponse =
                ssmClient.getParameter(parameterRequest);
        }
    }
}
```

```
        System.out.println("The parameter value is " +
parameterResponse.parameter().value());

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DescribeParameters](#) di Referensi AWS SDK for Java 2.x API.

## Memperbarui OpsItem

Contoh kode berikut menunjukkan cara memperbarui sebuah OpsItem.

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.SsmException;
import software.amazon.awssdk.services.ssm.model.UpdateOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.OpsItemStatus;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ResolveOpsItem {
    public static void main(String[] args) {
```



```
final String usage = ""

    Usage:
        <opsID>

    Where:
        opsID - The Ops item ID value.
    """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String opsID = args[0];
Region region = Region.US_EAST_1;
SsmClient ssmClient = SsmClient.builder()
    .region(region)
    .build();
setOpsItemStatus(ssmClient, opsID);
}

public static void setOpsItemStatus(SsmClient ssmClient, String opsID) {
    try {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
            .opsItemId(opsID)
            .status(OpsItemStatus.RESOLVED)
            .build();

        ssmClient.updateOpsItem(opsItemRequest);

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [UpdateOpsItem](#) di Referensi AWS SDK for Java 2.x API.

## Contoh Amazon Texttract menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan Amazon Texttract. AWS SDK for Java 2.x

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

- [Tindakan](#)

### Tindakan

Menganalisis dokumen

Contoh kode berikut menunjukkan cara menganalisis dokumen menggunakan Amazon Texttract.

SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.texttract.TexttractClient;
import software.amazon.awssdk.services.texttract.model.AnalyzeDocumentRequest;
import software.amazon.awssdk.services.texttract.model.Document;
import software.amazon.awssdk.services.texttract.model.FeatureType;
import software.amazon.awssdk.services.texttract.model.AnalyzeDocumentResponse;
import software.amazon.awssdk.services.texttract.model.Block;
import software.amazon.awssdk.services.texttract.model.TexttractException;
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AnalyzeDocument {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <sourceDoc>\s

            Where:
                sourceDoc - The path where the document is located (must be an
image, for example, C:/AWS/book.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceDoc = args[0];
        Region region = Region.US_EAST_2;
        TextractClient textractClient = TextractClient.builder()
            .region(region)
            .build();

        analyzeDoc(textractClient, sourceDoc);
        textractClient.close();
    }

    public static void analyzeDoc(TextractClient textractClient, String sourceDoc) {
```

```
try {
    InputStream sourceStream = new FileInputStream(new File(sourceDoc));
    SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

    // Get the input Document object as bytes
    Document myDoc = Document.builder()
        .bytes(sourceBytes)
        .build();

    List<FeatureType> featureTypes = new ArrayList<FeatureType>();
    featureTypes.add(FeatureType.FORMS);
    featureTypes.add(FeatureType.TABLES);

    AnalyzeDocumentRequest analyzeDocumentRequest =
AnalyzeDocumentRequest.builder()
        .featureTypes(featureTypes)
        .document(myDoc)
        .build();

    AnalyzeDocumentResponse analyzeDocument =
textractClient.analyzeDocument(analyzeDocumentRequest);
    List<Block> docInfo = analyzeDocument.blocks();
    Iterator<Block> blockIterator = docInfo.iterator();

    while (blockIterator.hasNext()) {
        Block block = blockIterator.next();
        System.out.println("The block type is " +
block.blockType().toString());
    }

} catch (TextractException | FileNotFoundException e) {

    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- Untuk detail API, lihat [AnalyzeDocument](#) di Referensi AWS SDK for Java 2.x API.

## Mendeteksi teks dalam dokumen

Contoh kode berikut menunjukkan cara mendeteksi teks dalam dokumen menggunakan Amazon Textract.

SDK for Java 2.x

### Note

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

Mendeteksi teks dari dokumen input.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectDocumentText {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
```

```

        <sourceDoc>\s

        Where:
            sourceDoc - The path where the document is located (must be an
image, for example, C:/AWS/book.png).\s
            """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String sourceDoc = args[0];
    Region region = Region.US_EAST_2;
    TextractClient textractClient = TextractClient.builder()
        .region(region)
        .build();

    detectDocText(textractClient, sourceDoc);
    textractClient.close();
}

public static void detectDocText(TextractClient textractClient, String
sourceDoc) {
    try {
        InputStream sourceStream = new FileInputStream(new File(sourceDoc));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Get the input Document object as bytes.
        Document myDoc = Document.builder()
            .bytes(sourceBytes)
            .build();

        DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
            .document(myDoc)
            .build();

        // Invoke the Detect operation.
        DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);
        List<Block> docInfo = textResponse.blocks();
        for (Block block : docInfo) {

```

```

        System.out.println("The block type is " +
block.blockType().toString());
    }

    DocumentMetadata documentMetadata = textResponse.documentMetadata();
    System.out.println("The number of pages in the document is " +
documentMetadata.pages());

    } catch (TextractException | FileNotFoundException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}

```

Mendeteksi teks dari dokumen yang terletak di bucket Amazon S3.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectDocumentTextS3 {

    public static void main(String[] args) {
        final String usage = ""

        Usage:

```

```

        <bucketName> <docName>\s

        Where:
            bucketName - The name of the Amazon S3 bucket that contains the
document.\s

            docName - The document name (must be an image, i.e., book.png).
\s

        """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
            .region(region)
            .build();

        detectDocTextS3(textractClient, bucketName, docName);
        textractClient.close();
    }

    public static void detectDocTextS3(TextractClient textractClient, String
bucketName, String docName) {
        try {
            S3Object s3Object = S3Object.builder()
                .bucket(bucketName)
                .name(docName)
                .build();

            // Create a Document object and reference the s3Object instance.
            Document myDoc = Document.builder()
                .s3Object(s3Object)
                .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
                .document(myDoc)
                .build();

```



```
        DetectDocumentTextResponse textResponse =
textextractClient.detectDocumentText(detectDocumentTextRequest);
        for (Block block : textResponse.blocks()) {
            System.out.println("The block type is " +
block.blockType().toString());
        }

        DocumentMetadata documentMetadata = textResponse.documentMetadata();
        System.out.println("The number of pages in the document is " +
documentMetadata.pages());

    } catch (TextractException e) {


        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DetectDocumentText](#) di Referensi AWS SDK for Java 2.x API.

Mulai analisis asinkron dokumen

Contoh kode berikut menunjukkan cara memulai analisis asinkron dokumen menggunakan Amazon Textract.

SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.DocumentLocation;
import software.amazon.awssdk.services.textract.model.TextractException;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisResponse;
```

```
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.FeatureType;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StartDocumentAnalysis {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <docName>\s

                Where:
                bucketName - The name of the Amazon S3 bucket that contains the
document.\s
                docName - The document name (must be an image, for example,
book.png).\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        String jobId = startDocAnalysisS3(textractClient, bucketName, docName);
        System.out.println("Getting results for job " + jobId);
        String status = getJobResults(textractClient, jobId);
        System.out.println("The job status is " + status);
    }
}
```

```
        textractClient.close();
    }

    public static String startDocAnalysisS3(TextractClient textractClient, String
bucketName, String docName) {
    try {
        List<FeatureType> myList = new ArrayList<>();
        myList.add(FeatureType.TABLES);
        myList.add(FeatureType.FORMS);

        S3Object s3Object = S3Object.builder()
            .bucket(bucketName)
            .name(docName)
            .build();

        DocumentLocation location = DocumentLocation.builder()
            .s3Object(s3Object)
            .build();

        StartDocumentAnalysisRequest documentAnalysisRequest =
StartDocumentAnalysisRequest.builder()
            .documentLocation(location)
            .featureTypes(myList)
            .build();

        StartDocumentAnalysisResponse response =
textractClient.startDocumentAnalysis(documentAnalysisRequest);

        // Get the job ID
        String jobId = response.jobId();
        return jobId;

    } catch (TextractException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

private static String getJobResults(TextractClient textractClient, String jobId)
{
    boolean finished = false;
    int index = 0;
    String status = "";
```

```
    try {
        while (!finished) {
            GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
                .jobId(jobId)
                .maxResults(1000)
                .build();

            GetDocumentAnalysisResponse response =
textractClient.getDocumentAnalysis(analysisRequest);
            status = response.jobStatus().toString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(index + " status is: " + status);
                Thread.sleep(1000);
            }
            index++;
        }

        return status;
    } catch (InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Untuk detail API, lihat [StartDocumentAnalysis](#) di Referensi AWS SDK for Java 2.x API.

## Contoh Amazon Transcribe menggunakan SDK for Java 2.x

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum menggunakan AWS SDK for Java 2.x with Amazon Transcribe.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

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

Topik

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

## Tindakan

Daftar pekerjaan transkripsi

Contoh kode berikut menunjukkan cara membuat daftar pekerjaan transkripsi Amazon Transcribe.

SDK for Java 2.x

### Note

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

```
public class ListTranscriptionJobs {
    public static void main(String[] args) {
        TranscribeClient transcribeClient = TranscribeClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listTranscriptionJobs(transcribeClient);
    }

    public static void listTranscriptionJobs(TranscribeClient transcribeClient)
    {
```

```

        ListTranscriptionJobsRequest listJobsRequest =
ListTranscriptionJobsRequest.builder()
        .build();

transcribeClient.listTranscriptionJobsPaginator(listJobsRequest).stream()
        .flatMap(response -> response.transcriptionJobSummaries().stream())
        .forEach(jobSummary -> {
            System.out.println("Job Name: " +
jobSummary.transcriptionJobName());
            System.out.println("Job Status: " +
jobSummary.transcriptionJobStatus());
            System.out.println("Output Location: " +
jobSummary.outputLocationType());
            // Add more information as needed

            // Retrieve additional details for the job if necessary
            GetTranscriptionJobResponse jobDetails =
transcribeClient.getTranscriptionJob(
                GetTranscriptionJobRequest.builder()
                    .transcriptionJobName(jobSummary.transcriptionJobName())
                    .build());

            // Display additional details
            System.out.println("Language Code: " +
jobDetails.transcriptionJob().languageCode());
            System.out.println("Media Format: " +
jobDetails.transcriptionJob().mediaFormat());
            // Add more details as needed

            System.out.println("-----");
        });
    }
}

```

- Untuk detail API, lihat [ListTranscriptionJobs](#) di Referensi AWS SDK for Java 2.x API.

## Mulai pekerjaan transkripsi

Contoh kode berikut menunjukkan cara memulai pekerjaan transkripsi Amazon Transcribe.

## SDK for Java 2.x

 Note

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

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws URISyntaxException, ExecutionException, InterruptedException,
        LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .region(REGION)
            .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromMic()),
    getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }
    }
}
```

```

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
            .languageCode(LanguageCode.EN_US.toString())
            .mediaEncoding(MediaEncoding.PCM)
            .mediaSampleRateHertz(mediaSampleRateHertz)
            .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
            .onResponse(r -> {
                System.out.println("Received Initial response");
            })
            .onError(e -> {
                System.out.println(e.getMessage());
                StringWriter sw = new StringWriter();
                e.printStackTrace(new PrintWriter(sw));
                System.out.println("Error Occurred: " + sw.toString());
            })
            .onComplete(() -> {
                System.out.println("=== All records stream successfully ===");
            })
            .subscriber(event -> {
                List<Result> results = ((TranscriptEvent)
event).transcript().results();
                if (results.size() > 0) {
                    if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {
                        System.out.println(results.get(0).alternatives().get(0).transcript());
                    }
                }
            })
    }

```



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

private InputStream getStreamFromFile(String audioFileName) {
    try {
        File inputFile = new
File(getClass().getClassLoader().getResource(audioFileName).getFile());
        InputStream audioStream = new FileInputStream(inputFile);
        return audioStream;
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}

private static class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;
    private static Subscription currentSubscription;

    private AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {

        if (this.currentSubscription == null) {
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        } else {
            this.currentSubscription.cancel();
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        }
        s.onSubscribe(currentSubscription);
    }
}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);
}
```

```
SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
    this.subscriber = s;
    this.inputStream = inputStream;
}

@Override
public void request(long n) {
    if (n <= 0) {
        subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
    }

    demand.getAndAdd(n);

    executor.submit(() -> {
        try {
            do {
                ByteBuffer audioBuffer = getNextEvent();
                if (audioBuffer.remaining() > 0) {
                    AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                    subscriber.onNext(audioEvent);
                } else {
                    subscriber.onComplete();
                    break;
                }
            } while (demand.decrementAndGet() > 0);
        } catch (Exception e) {
            subscriber.onError(e);
        }
    });
}

@Override
public void cancel() {
    executor.shutdown();
}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer = null;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

    int len = 0;
```

```
        try {
            len = inputStream.read(audioBytes);

            if (len <= 0) {
                audioBuffer = ByteBuffer.allocate(0);
            } else {
                audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
            }
        } catch (IOException e) {
            throw new UncheckedIOException(e);
        }

        return audioBuffer;
    }

    private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
        return AudioEvent.builder()
            .audioChunk(SdkBytes.fromByteBuffer(bb))
            .build();
    }
}
```

- Untuk detail API, lihat [StartTranscriptionJob](#) di Referensi AWS SDK for Java 2.x API.

## Skenario

Transcribe audio dan dapatkan data pekerjaan

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Mulai pekerjaan transkripsi dengan Amazon Transcribe.
- Tunggu hingga tugas selesai.
- Dapatkan URI tempat transkrip disimpan.

Untuk informasi selengkapnya, lihat [Memulai Amazon Transcribe](#).

## SDK for Java 2.x

 Note

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

## Mentranskripsikan file PCM.

```
/**
 * To run this AWS code example, ensure that you have set up your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class TranscribeStreamingDemoFile {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[]) throws ExecutionException,
    InterruptedException {

        final String USAGE = "\n" +
            "Usage:\n" +
            "  <file> \n\n" +
            "Where:\n" +
            "  file - the location of a PCM file to transcribe. In this
example, ensure the PCM file is 16 hertz (Hz). \n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String file = args[0];
        client = TranscribeStreamingAsyncClient.builder()
            .region(REGION)
            .build();
    }
}
```

```
        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromFile(file)),
    getResponseHandler());

    result.get();
    client.close();
}

private static InputStream getStreamFromFile(String file) {
    try {
        File inputFile = new File(file);
        InputStream audioStream = new FileInputStream(inputFile);
        return audioStream;

    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}

private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
    return StartStreamTranscriptionRequest.builder()
        .languageCode(LanguageCode.EN_US)
        .mediaEncoding(MediaEncoding.PCM)
        .mediaSampleRateHertz(mediaSampleRateHertz)
        .build();
}

private static StartStreamTranscriptionResponseHandler getResponseHandler() {
    return StartStreamTranscriptionResponseHandler.builder()
        .onResponse(r -> {
            System.out.println("Received Initial response");
        })
        .onError(e -> {
            System.out.println(e.getMessage());
            StringWriter sw = new StringWriter();
            e.printStackTrace(new PrintWriter(sw));
            System.out.println("Error Occurred: " + sw.toString());
        })
        .onComplete(() -> {
            System.out.println("=== All records stream successfully ===");
        })
        .subscriber(event -> {
```

```

        List<Result> results = ((TranscriptEvent)
event).transcript().results();
        if (results.size() > 0) {
            if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

System.out.println(results.get(0).alternatives().get(0).transcript());
            }
        }
    })
    .build();
}

private static class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;
    private static Subscription currentSubscription;

    private AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {

        if (this.currentSubscription == null) {
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        } else {
            this.currentSubscription.cancel();
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        }
        s.onSubscribe(currentSubscription);
    }
}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);

    SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
        this.subscriber = s;

```

```
        this.inputStream = inputStream;
    }

    @Override
    public void request(long n) {
        if (n <= 0) {
            subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
        }

        demand.getAndAdd(n);

        executor.submit(() -> {
            try {
                do {
                    ByteBuffer audioBuffer = getNextEvent();
                    if (audioBuffer.remaining() > 0) {
                        AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                        subscriber.onNext(audioEvent);
                    } else {
                        subscriber.onComplete();
                        break;
                    }
                } while (demand.decrementAndGet() > 0);
            } catch (Exception e) {
                subscriber.onError(e);
            }
        });
    }

    @Override
    public void cancel() {
        executor.shutdown();
    }

    private ByteBuffer getNextEvent() {
        ByteBuffer audioBuffer = null;
        byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

        int len = 0;
        try {
            len = inputStream.read(audioBytes);
        }
```

```

        if (len <= 0) {
            audioBuffer = ByteBuffer.allocate(0);
        } else {
            audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
        }
    } catch (IOException e) {
        throw new UncheckedIOException(e);
    }

    return audioBuffer;
}

private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
}

```

Mentranskripsikan audio streaming dari mikrofon komputer Anda.

```

public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws URISyntaxException, ExecutionException, InterruptedException,
        LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .region(REGION)
            .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromMic()),
    getResponseHandler());

        result.get();
        client.close();
    }
}

```



```
}

private static InputStream getStreamFromMic() throws LineUnavailableException {

    // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
    int sampleRate = 16000;
    AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
    DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

    if (!AudioSystem.isLineSupported(info)) {
        System.out.println("Line not supported");
        System.exit(0);
    }

    TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
    line.open(format);
    line.start();

    InputStream audioStream = new AudioInputStream(line);
    return audioStream;
}

private static AwsCredentialsProvider getCredentials() {
    return DefaultCredentialsProvider.create();
}

private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
    return StartStreamTranscriptionRequest.builder()
        .languageCode(LanguageCode.EN_US.toString())
        .mediaEncoding(MediaEncoding.PCM)
        .mediaSampleRateHertz(mediaSampleRateHertz)
        .build();
}

private static StartStreamTranscriptionResponseHandler getResponseHandler() {
    return StartStreamTranscriptionResponseHandler.builder()
        .onResponse(r -> {
            System.out.println("Received Initial response");
        })
        .onError(e -> {
            System.out.println(e.getMessage());
            StringWriter sw = new StringWriter();
            e.printStackTrace(new PrintWriter(sw));
        });
}
```

```

        System.out.println("Error Occurred: " + sw.toString());
    })
    .onComplete(() -> {
        System.out.println("=== All records stream successfully ===");
    })
    .subscriber(event -> {
        List<Result> results = ((TranscriptEvent)
event).transcript().results();
        if (results.size() > 0) {
            if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

System.out.println(results.get(0).alternatives().get(0).transcript());
                }
            }
        })
    .build();
}

private InputStream getStreamFromFile(String audioFileName) {
    try {
        File inputFile = new
File(getClass().getClassLoader().getResource(audioFileName).getFile());
        InputStream audioStream = new FileInputStream(inputFile);
        return audioStream;
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}

private static class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;
    private static Subscription currentSubscription;

    private AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {

        if (this.currentSubscription == null) {
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        } else {

```

```

        this.currentSubscription.cancel();
        this.currentSubscription = new SubscriptionImpl(s, inputStream);
    }
    s.onSubscribe(currentSubscription);
}
}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);

    SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
    {
        this.subscriber = s;
        this.inputStream = inputStream;
    }

    @Override
    public void request(long n) {
        if (n <= 0) {
            subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
        }

        demand.getAndAdd(n);

        executor.submit(() -> {
            try {
                do {
                    ByteBuffer audioBuffer = getNextEvent();
                    if (audioBuffer.remaining() > 0) {
                        AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                        subscriber.onNext(audioEvent);
                    } else {
                        subscriber.onComplete();
                        break;
                    }
                } while (demand.decrementAndGet() > 0);
            } catch (Exception e) {
                subscriber.onError(e);
            }
        });
    }
}

```

```
        }
    });
}

@Override
public void cancel() {
    executor.shutdown();
}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer = null;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

    int len = 0;
    try {
        len = inputStream.read(audioBytes);

        if (len <= 0) {
            audioBuffer = ByteBuffer.allocate(0);
        } else {
            audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
        }
    } catch (IOException e) {
        throw new UncheckedIOException(e);
    }

    return audioBuffer;
}

private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - [GetTranscriptionJob](#)
  - [StartTranscriptionJob](#)

## Contoh lintas layanan menggunakan SDK for Java 2.x

Contoh aplikasi berikut menggunakan AWS SDK for Java 2.x untuk bekerja di beberapa Layanan AWS.

Contoh lintas layanan menargetkan pengalaman tingkat lanjut untuk membantu Anda mulai membangun aplikasi.

Contoh-contoh

- [Membangun aplikasi untuk mengirimkan data ke tabel DynamoDB](#)
- [Buat chatbot Amazon Lex untuk melibatkan pengunjung situs web Anda](#)
- [Membangun aplikasi terbitan dan berlangganan yang menerjemahkan pesan](#)
- [Buat aplikasi web yang mengirim dan mengambil pesan dengan menggunakan Amazon SQS](#)
- [Membuat aplikasi manajemen aset foto yang memungkinkan pengguna mengelola foto menggunakan label](#)
- [Membuat aplikasi web untuk melacak data DynamoDB](#)
- [Buat pelacak item Amazon Redshift](#)
- [Buat pelacak butir kerja Aurora Serverless](#)
- [Buat aplikasi yang menganalisis umpan balik pelanggan dan mensintesis audio](#)
- [Mendeteksi APD dalam gambar dengan Amazon AWS Rekognition menggunakan SDK](#)
- [Mendeteksi objek dalam gambar dengan Amazon Rekognition menggunakan SDK AWS](#)
- [Mendeteksi orang dan objek dalam video dengan Amazon Rekognition menggunakan SDK AWS](#)
- [Mempublikasikan pesan Amazon SNS ke antrian Amazon SQS menggunakan SDK AWS](#)
- [Menggunakan API Gateway untuk menginvokasi fungsi Lambda](#)
- [Menggunakan Step Functions untuk menginvokasi fungsi Lambda](#)
- [Menggunakan peristiwa terjadwal untuk menginvokasi fungsi Lambda](#)

## Membangun aplikasi untuk mengirimkan data ke tabel DynamoDB

SDK for Java 2.x

Menunjukkan cara membuat aplikasi web dinamis yang mengirimkan data menggunakan API Java Amazon DynamoDB dan mengirim pesan teks menggunakan API Java Amazon Simple Notification Service.

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

## Buat chatbot Amazon Lex untuk melibatkan pengunjung situs web Anda

SDK for Java 2.x

Menunjukkan cara menggunakan Amazon Lex API untuk membuat Chatbot dalam aplikasi web untuk melibatkan pengunjung situs web Anda.

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 Lex
- Amazon Translate

## Membangun aplikasi terbitkan dan berlangganan yang menerjemahkan pesan

SDK for Java 2.x

Menunjukkan cara menggunakan Amazon Simple Notification Service Java API untuk membuat aplikasi web yang memiliki fungsi berlangganan dan mempublikasikan. Selain itu, contoh aplikasi ini juga menerjemahkan pesan.

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

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan contoh yang menggunakan Java Async API, lihat contoh lengkapnya di [GitHub](#)

Layanan yang digunakan dalam contoh ini

- Amazon SNS
- Amazon Translate

## Buat aplikasi web yang mengirim dan mengambil pesan dengan menggunakan Amazon SQS

SDK for Java 2.x

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

## Membuat aplikasi manajemen aset foto yang memungkinkan pengguna mengelola foto menggunakan label

SDK for Java 2.x

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 posting 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

### SDK for Java 2.x

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 pelacak item Amazon Redshift

### SDK for Java 2.x

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 Serverless

### SDK for Java 2.x

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](#)

Untuk kode sumber lengkap dan instruksi tentang cara menyiapkan dan menjalankan contoh yang menggunakan JDBC API, lihat contoh lengkapnya di [GitHub](#)

Layanan yang digunakan dalam contoh ini

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

## Buat aplikasi yang menganalisis umpan balik pelanggan dan mensintesis audio

SDK for Java 2.x

Aplikasi contoh ini menganalisis dan menyimpan kartu umpan balik pelanggan. Secara khusus, ini memenuhi kebutuhan hotel fiktif di New York City. Hotel menerima umpan balik dari para tamu dalam berbagai bahasa dalam bentuk kartu komentar fisik. Umpan balik itu diunggah ke aplikasi melalui klien web. Setelah gambar kartu komentar diunggah, langkah-langkah berikut terjadi:

- Teks diekstraksi dari gambar menggunakan Amazon Textract.
- Amazon Comprehend menentukan sentimen teks yang diekstraksi dan bahasanya.
- Teks yang diekstraksi diterjemahkan ke bahasa Inggris menggunakan Amazon Translate.
- Amazon Polly mensintesis file audio dari teks yang diekstraksi.

Aplikasi lengkap dapat digunakan dengan AWS CDK Untuk kode sumber dan petunjuk penerapan, lihat proyek di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Comprehend
- Lambda
- Amazon Polly

- Amazon Textract
- Amazon Translate

## Mendeteksi APD dalam gambar dengan Amazon AWS Rekognition menggunakan SDK

### SDK for Java 2.x

Menunjukkan cara membuat AWS Lambda fungsi yang mendeteksi gambar dengan Alat Pelindung Diri.

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

Layanan yang digunakan dalam contoh ini

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

## Mendeteksi objek dalam gambar dengan Amazon Rekognition menggunakan SDK AWS

### SDK for Java 2.x

Menunjukkan cara menggunakan Amazon Rekognition Java API untuk membuat aplikasi yang menggunakan Amazon Rekognition untuk mengidentifikasi objek berdasarkan kategori dalam gambar yang terletak 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

## Mendeteksi orang dan objek dalam video dengan Amazon Rekognition menggunakan SDK AWS

### SDK for Java 2.x

Menunjukkan cara menggunakan Amazon Rekognition Java API untuk membuat aplikasi guna mendeteksi wajah dan objek di video 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

## Mempublikasikan pesan Amazon SNS ke antrian Amazon SQS menggunakan SDK AWS

### SDK for Java 2.x

Mendemonstrasikan pesan dengan topik dan antrian menggunakan Amazon Simple Notification Service (Amazon SNS) dan Amazon Simple Queue Service (Amazon SQS).

Untuk kode sumber lengkap dan instruksi yang menunjukkan pesan dengan topik dan antrian di Amazon SNS dan Amazon SQS, lihat contoh lengkapnya di [GitHub](#)

Layanan yang digunakan dalam contoh ini

- Amazon SNS
- Amazon SQS

## Menggunakan API Gateway untuk menginvokasi fungsi Lambda

### SDK for Java 2.x

Menunjukkan cara membuat AWS Lambda fungsi dengan menggunakan Lambda Java runtime API. Contoh ini memanggil AWS layanan yang berbeda untuk melakukan kasus penggunaan tertentu. Contoh ini menunjukkan cara membuat fungsi Lambda yang diinvokasi oleh Amazon API Gateway yang memindai peringatan hari jadi kerja di tabel Amazon DynamoDB dan menggunakan Amazon Simple Notification Service (Amazon SNS) untuk mengirim pesan teks berisi ucapan selamat kepada karyawan Anda pada tanggal hari jadi kerja satu tahun mereka.

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

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

## Menggunakan Step Functions untuk menginvokasi fungsi Lambda

### SDK for Java 2.x

Menunjukkan cara membuat alur kerja AWS tanpa server dengan menggunakan AWS Step Functions dan. AWS SDK for Java 2.x Setiap langkah alur kerja diimplementasikan menggunakan AWS Lambda fungsi.

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

Layanan yang digunakan dalam contoh ini

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

# Menggunakan peristiwa terjadwal untuk menginvokasi fungsi Lambda

## SDK for Java 2.x

Menunjukkan cara membuat acara EventBridge terjadwal Amazon yang memanggil AWS Lambda fungsi. Konfigurasi EventBridge untuk menggunakan ekspresi cron untuk menjadwalkan saat fungsi Lambda dipanggil. Dalam contoh ini, Anda membuat fungsi Lambda menggunakan API runtime Java Lambda. Contoh ini memanggil AWS layanan yang berbeda untuk melakukan kasus penggunaan tertentu. Contoh ini menunjukkan cara membuat aplikasi yang mengirimkan pesan teks seluler kepada karyawan Anda berisi ucapan selamat pada hari jadi setahun kerja mereka.

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

Layanan yang digunakan dalam contoh ini

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

# Keamanan untuk AWS SDK for Java

Keamanan cloud di Amazon Web Services (AWS) merupakan prioritas tertinggi. Sebagai pelanggan AWS, Anda mendapatkan manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan 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 dalam AWS SDK for Java 2.x](#)
- [Bekerja dengan TLS di SDK for Java](#)
- [Manajemen Identitas dan Akses](#)
- [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 dalam AWS SDK for Java 2.x

[Model tanggung jawab AWS bersama model](#) berlaku untuk perlindungan data di AWS SDK for Java. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur

global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk memelihara kendali atas isi yang dihost 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.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi yang lebih lengkap tentang titik akhir FIPS yang tersedia di [Standar Pemrosesan Informasi Federal \(FIPS\) 140-2](#).

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 for Java atau Layanan AWS lainnya menggunakan konsol, API AWS CLI, AWS atau SDK. 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.

# Bekerja dengan TLS di SDK for Java

AWS SDK for Java Menggunakan kemampuan TLS dari platform Java yang mendasarinya. [Dalam topik ini, kami menunjukkan contoh menggunakan implementasi OpenJDK yang digunakan oleh Amazon Corretto 17.](#)

Untuk bekerja dengan Layanan AWS, JDK yang mendasarinya harus mendukung versi minimum TLS 1.2, tetapi TLS 1.3 direkomendasikan.

Pengguna harus berkonsultasi dengan dokumentasi platform Java yang mereka gunakan dengan SDK untuk mengetahui versi TLS mana yang diaktifkan secara default serta cara mengaktifkan dan menonaktifkan versi TLS tertentu.

## Cara memeriksa informasi versi TLS

Menggunakan OpenJDK, kode berikut menunjukkan penggunaan SSLContext untuk mencetak versi TLS/SSL mana [yang](#) didukung.

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProtocols()));
```

Misalnya, Amazon Corretto 17 (OpenJDK) menghasilkan output berikut.

```
[TLSv1.3, TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2Hello]
```

Untuk melihat jabat tangan SSL beraksi dan versi TLS apa yang digunakan, Anda dapat menggunakan properti sistem `javax.net.debug`.

Misalnya, jalankan aplikasi Java yang menggunakan TLS.

```
java app.jar -Djavax.net.debug=ssl:handshake
```

Aplikasi mencatat jabat tangan SSL yang mirip dengan yang berikut ini.

```
...
javax.net.ssl|DEBUG|10|main|2022-12-23 13:53:12.221 EST|ClientHello.java:641|Produced
ClientHello handshake message (
"ClientHello": {
  "client version"      : "TLSv1.2",
```



```
...
javax.net.ssl|DEBUG|10|main|2022-12-23 13:53:12.295 EST|ServerHello.java:888|Consuming
ServerHello handshake message (
"ServerHello": {
  "server version"      : "TLSv1.2",
...

```

## Menerapkan versi TLS minimum

SDK for Java selalu lebih menyukai versi TLS terbaru yang didukung oleh platform dan layanan. Jika Anda ingin menerapkan versi TLS minimum tertentu, lihat dokumentasi platform Java Anda.

Untuk JVM berbasis OpenJDK, Anda dapat menggunakan properti sistem.

```
jdk.tls.client.protocols
```

Misalnya, jika Anda ingin klien layanan SDK dalam aplikasi Anda menggunakan TLS 1.2, meskipun TLS 1.3 tersedia, berikan properti sistem berikut.

```
java app.jar -Djdk.tls.client.protocols=TLSv1.2
```

## AWS Upgrade titik akhir API ke TLS 1.2

Lihat [posting blog](#) ini untuk informasi tentang titik akhir AWS API yang pindah ke TLS 1.2 untuk versi minimum.

## Manajemen Identitas dan Akses

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 Anda untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep Basic 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 kredensial 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 gabungan, 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. Untuk informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat [Menandatangani permintaan AWS API](#) di 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 lebih lanjut, lihat [Autentikasi multi-faktor](#) dalam Panduan Pengguna AWS IAM Identity Center dan [Menggunakan autentikasi multi-faktor \(MFA\) di AWS](#) 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 Anda. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar tugas lengkap yang mengharuskan Anda masuk sebagai pengguna root, lihat [Tugas yang memerlukan kredensial pengguna root](#) dalam Panduan Pengguna IAM.

## Identitas terfederasi

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 pengelolaan akses terpusat, sebaiknya 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 [Apa yang dimaksud 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, sebaiknya andalkan kredensial temporer, dan bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensial jangka panjang dengan pengguna IAM, sebaiknya rotasikan 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 kumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin untuk beberapa pengguna sekaligus. Grup membuat izin lebih mudah dikelola untuk sekelompok besar pengguna. Misalnya, Anda dapat memiliki grup yang bernama 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 tersebut dimaksudkan untuk dapat diambil oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, silakan lihat [Kapan harus membuat pengguna IAM \(bukan peran\)](#) 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. Anda dapat mengambil peran IAM untuk sementara AWS Management Console dengan [beralih peran](#). Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang metode untuk menggunakan peran, lihat [Menggunakan peran IAM](#) dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna gabungan – Untuk menetapkan izin ke sebuah identitas gabungan, Anda dapat membuat peran dan menentukan izin untuk peran tersebut. Saat identitas terfederasi diautentikasi, identitas tersebut dikaitkan dengan peran dan diberikan izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat [Membuat peran untuk Penyedia Identitas pihak ketiga](#) dalam Panduan Pengguna IAM. Jika Anda menggunakan Pusat Identitas IAM, Anda mengonfigurasi sekumpulan izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM mengaitkan izin yang ditetapkan ke peran dalam IAM. Untuk informasi tentang rangkaian izin, lihat [Rangkaian 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 (pengguna utama tepercaya) dengan akun berbeda untuk mengakses sumber daya yang ada 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 kebijakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya](#) dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Contoh, ketika Anda melakukan panggilan dalam layanan, umumnya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Suatu layanan mungkin melakukan hal tersebut menggunakan izin pengguna utama panggilan, 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 tindakan yang kemudian memulai 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 [Meneruskan sesi akses](#).
- Peran IAM – Peran layanan adalah [peran IAM](#) yang diambil layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus

peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat [Membuat peran untuk mendelegasikan izin ke Layanan AWS](#) dalam Panduan pengguna IAM.

- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. Layanan AWS Layanan tersebut dapat mengambil peran untuk melakukan sebuah 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 kredensial sementara untuk aplikasi yang berjalan pada instans EC2 dan membuat atau permintaan API. AWS CLI AWS Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan AWS peran ke instans EC2 dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat [Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di instans Amazon EC2](#) dalam Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat [Kapan harus membuat peran IAM \(bukan pengguna\)](#) dalam 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 [Ikhtisar kebijakan JSON](#) dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat menjalankan peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk operasi. Sebagai contoh, anggap saja 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 pengguna dan peran, di sumber daya mana, dan dengan ketentuan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat [Membuat kebijakan IAM](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan terkelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan 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 terkelola atau kebijakan inline, lihat [Memilih antara kebijakan terkelola 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 yang dilampiri kebijakan tersebut, kebijakan ini menentukan jenis tindakan yang dapat dilakukan oleh pengguna utama tertentu di sumber daya tersebut dan apa ketentuannya. Anda harus [menentukan pengguna utama](#) 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 (ACL)

Daftar kontrol akses (ACL) mengendalikan pengguna utama mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL sama dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, silakan lihat [Gambaran umum daftar kontrol akses \(ACL\)](#) di Panduan Developer Layanan Penyimpanan Ringkas Amazon.

## Tipe kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Tipe-tipe kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda berdasarkan tipe kebijakan yang lebih umum.

- **Batasan izin** – Batasan izin adalah fitur lanjutan di mana Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM (pengguna 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 secara eksplisit terhadap salah satu kebijakan ini akan mengesampingkan izin tersebut. Untuk informasi selengkapnya tentang batasan izin, lihat [Batasan izin untuk entitas IAM](#) dalam Panduan Pengguna IAM.
- **Kebijakan kontrol layanan (SCP)** — SCP 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 organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke sebagian atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS Untuk informasi selengkapnya tentang Organisasi dan SCP, lihat [Cara kerja SCP](#) dalam Panduan Pengguna AWS Organizations .
- **Kebijakan sesi** – Kebijakan sesi adalah kebijakan lanjutan yang Anda teruskan sebagai parameter saat Anda membuat sesi sementara secara terprogram untuk peran atau pengguna gabungan. 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 di salah satu kebijakan ini akan membatalkan izin tersebut. Untuk informasi selengkapnya, lihat [Kebijakan sesi](#) dalam Panduan Pengguna IAM.

## Berbagai jenis kebijakan

Jika beberapa jenis kebijakan diberlakukan untuk satu 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 bekerja 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 pengguna 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 (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi pengguna akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa hal berikut:

- Untuk mempelajari apakah AWS mendukung 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 [Memberikan akses kepada pengguna eksternal yang sah \(federasi identitas\)](#) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara penggunaan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Perbedaan antara peran IAM dan kebijakan berbasis sumber daya](#) 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:

- [Panduan Memulai Cepat Keamanan dan Kepatuhan — Panduan](#) penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar AWS yang berfokus pada keamanan dan kepatuhan.
- [Arsitektur untuk Keamanan dan Kepatuhan HIPAA di Amazon Web Services](#) — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan AWS untuk membuat aplikasi yang memenuhi syarat HIPAA.

### Note

Tidak semua memenuhi Layanan AWS syarat HIPAA. Untuk informasi selengkapnya, lihat [Referensi Layanan yang Memenuhi 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 mengevaluasi sumber daya AWS Anda dan memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat [Referensi kontrol Security Hub](#).
- [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 dengan menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan pengguna utama 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](#).

# Migrasi dari versi 1.x ke 2.x dari AWS SDK for Java

AWS SDK for Java 2.x adalah penulisan ulang utama dari basis kode 1.x yang dibangun di atas Java 8+. Ini mencakup banyak pembaruan, seperti peningkatan konsistensi, kemudahan penggunaan, dan kekekalan yang ditegakkan dengan kuat. Bagian ini menjelaskan fitur utama yang baru di versi 2.x, dan memberikan panduan tentang cara memigrasikan kode Anda ke versi 2.x dari 1.x.

## Topik

- [Apa yang baru di versi 2](#)
- [step-by-step Instruksi migrasi dengan contoh](#)
- [Apa yang berbeda antara AWS SDK for Java 1.x dan 2.x](#)
- [Gunakan SDK for Java 1.x and 2.x side-by-side](#)

## Apa yang baru di versi 2

- Anda dapat mengkonfigurasi klien HTTP Anda sendiri. Lihat [konfigurasi transport HTTP](#).
- Klien async menampilkan dukungan I/O yang tidak memblokir dan mengembalikan objek `CompletableFuture` Lihat [Pemrograman asinkron](#).
- Operasi yang mengembalikan beberapa halaman memiliki respons autopaginasi. Dengan cara ini, Anda dapat memfokuskan kode Anda pada apa yang harus dilakukan dengan respons, tanpa perlu memeriksa dan mendapatkan halaman berikutnya. Lihat [Pagination](#).
- Kinerja waktu mulai SDK untuk AWS Lambda fungsi ditingkatkan. Lihat [Peningkatan kinerja waktu mulai SDK](#).
- Versi 2.x mendukung metode singkatan baru untuk membuat permintaan.

## Example

```
dynamoDbClient.putItem(request -> request.tableName(TABLE))
```

Untuk detail lebih lanjut tentang fitur baru dan untuk melihat contoh kode tertentu, lihat bagian lain dari panduan ini.

- [Mulai Cepat](#)
- [Menyiapkan](#)

- [Contoh kode untuk AWS SDK for Java 2.x](#)
- [Gunakan SDK](#)
- [Keamanan untuk AWS SDK for Java](#)

## step-by-step Instruksi migrasi dengan contoh

Bagian ini menyediakan step-by-step panduan untuk memigrasikan aplikasi Anda yang saat ini menggunakan SDK for Java v1.x ke SDK for Java 2.x. Bagian pertama menyajikan ikhtisar langkah-langkah diikuti dengan contoh rinci migrasi.

Langkah-langkah yang dibahas di sini menjelaskan migrasi kasus penggunaan normal, di mana aplikasi memanggil Layanan AWS menggunakan klien layanan berbasis model. Jika Anda perlu memigrasikan kode yang menggunakan API tingkat yang lebih tinggi seperti [S3 Transfer Manager](#) atau [CloudFrontpresigning](#), lihat bagian di bawah [the section called “Apa yang berbeda antara 1.x dan 2.x”](#) daftar isi.

Pendekatan yang dijelaskan di sini adalah saran. Anda dapat menggunakan teknik lain dan memanfaatkan fitur pengeditan kode IDE Anda untuk mencapai hasil yang sama.

## Ikhtisar langkah-langkah

### 1. Mulailah dengan menambahkan SDK for Java 2.x BOM

Dengan menambahkan elemen Maven BOM (Bill of Materials) untuk SDK for Java 2.x ke file POM Anda, Anda memastikan bahwa semua ketergantungan v2 yang Anda butuhkan berasal dari versi yang sama. POM Anda dapat berisi dependensi v1 dan v2. Ini memungkinkan Anda untuk memigrasikan kode secara bertahap daripada mengubah semuanya sekaligus.

#### SDK for Java 2.x BOM

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.24.3</version>
      <type>pom</type>
```

```
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
```

Anda dapat menemukan [versi terbaru](#) di Maven Central Repository.

## 2. Cari file untuk pernyataan impor kelas v1

Dengan memindai file dalam aplikasi Anda untuk impor v1, Anda akan menemukan dependensi v2 untuk ditambahkan ke file POM Maven Anda.

## 3. Tentukan dependensi Maven v2 dari pernyataan impor v1

Setelah Anda menemukan semua pernyataan impor v1 yang unik, Anda dapat menentukan artefak Maven yang sesuai untuk ketergantungan v2 dengan merujuk ke nama Package to dependency mapping table.

## 4. Tambahkan elemen ketergantungan v2 ke file POM

Perbarui file Maven POM dengan elemen ketergantungan ditentukan pada langkah 3.

## 5. Dalam file Java, secara bertahap ubah kelas v1 ke kelas v2

Saat Anda melakukan ini, buat perubahan yang diperlukan untuk mendukung API v2 seperti menggunakan pembangun alih-alih konstruktor dan menggunakan pengambil dan penyetel yang lancar.

## 6. Hapus dependensi v1 Maven dari impor POM dan v1 dari file

Saat Anda melakukan ini, buat perubahan yang diperlukan untuk mendukung API v2 seperti menggunakan pembangun alih-alih konstruktor dan menggunakan pengambil dan penyetel yang lancar.

## 7. Memfaktorkan ulang kode untuk menggunakan penyempurnaan API v2

Setelah kode berhasil dikompilasi sebagai lulus tes, Anda dapat memanfaatkan penyempurnaan v2 seperti menggunakan klien HTTP atau paginator yang berbeda untuk menyederhanakan kode. Ini adalah langkah opsional.



## Contoh migrasi

Dalam contoh ini, kami memigrasikan aplikasi yang menggunakan SDK for Java v1 dan mengakses beberapa. Layanan AWS Kami bekerja melalui metode v1 berikut secara rinci di langkah 5. Ini adalah salah satu metode dalam kelas yang berisi delapan metode dan ada 32 kelas dalam aplikasi.

metode v1 untuk bermigrasi

Hanya impor SDK v1 yang tercantum di bawah ini dari file Java.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesResult;
import com.amazonaws.services.ec2.model.Instance;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Reservation;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;
...
private static List<Instance> getRunningInstances(AmazonEC2Client ec2, List<String>
instanceIds) {
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = new DescribeInstancesRequest()
            .withInstanceIds(instanceIds);
        DescribeInstancesResult result;
        do {
            // DescribeInstancesResponse is a paginated response, so use tokens with
multiple requests.
            result = ec2.describeInstances(request);
            request.setNextToken(result.getNextToken()); // Prepare request for next
page.
            for (final Reservation r : result.getReservations()) {
                for (final Instance instance : r.getInstances()) {
                    LOGGER.info("Examining instanceId: " + instance.getInstanceId());
                    // if instance is in a running state, add it to runningInstances
list.
                    if (RUNNING_STATES.contains(instance.getState().getName())) {
```

```

        runningInstances.add(instance);
    }
}
} while (result.getNextToken() != null);
} catch (final AmazonEC2Exception exception) {
    // if instance isn't found, assume its terminated and continue.
    if (exception.getErrorCode().equals(NOT_FOUND_ERROR_CODE)) {
        LOGGER.info("Instance probably terminated; moving on.");
    } else {
        throw exception;
    }
}
return runningInstances;
}

```

## 1. Tambahkan v2 Maven BOM

Tambahkan Maven BOM untuk SDK for Java 2.x ke POM di samping dependensi lain di bagian tersebut. `dependencyManagement` Jika file POM Anda memiliki BOM untuk v1 SDK, biarkan untuk saat ini. Ini akan dihapus pada langkah selanjutnya.

### Manajemen Ketergantungan POM di awal

```

<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.example</groupId>           <!--Existing dependency in POM. -->
      <artifactId>bom</artifactId>
      <version>1.3.4</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    ...
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId> <!--Existing v1 BOM dependency. -->
      <version>1.11.1000</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    ...
  </dependencies>
</dependencyManagement>

```

```
<groupId>software.amazon.awssdk</groupId> <!--Add v2 BOM dependency. -->
<artifactId>bom</artifactId>
<version>2.24.3</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
```

## 2. Cari file untuk pernyataan impor kelas v1

Cari kode aplikasi untuk kejadian unik. `import com.amazonaws.services` Ini membantu kami menentukan dependensi v1 yang digunakan oleh proyek. Jika aplikasi Anda memiliki file POM Maven dengan dependensi v1 terdaftar, Anda dapat menggunakan informasi ini sebagai gantinya.

Untuk contoh ini kita menggunakan perintah [ripgrep\(rg\)](#) untuk mencari basis kode.

Dari root basis kode Anda, jalankan `ripgrep` perintah berikut. Setelah `ripgrep` menemukan pernyataan impor, mereka disalurkan ke `cut`, `sort`, dan `uniq` perintah untuk mengisolasi nama layanan.

```
rg --no-filename 'import\s+com\.amazonaws\.services' | cut -d '.' -f 4 | sort | uniq
```

Untuk aplikasi ini, berikut ini dicatat ke konsol.

```
autoscaling
cloudformation
ec2
identitymanagement
```

Ini menunjukkan bahwa setidaknya ada satu kemunculan dari masing-masing nama paket berikut yang digunakan dalam `import` pernyataan. Empat tujuan kita, nama kelas individu tidak masalah. Kami hanya perlu menemukan layanan yang digunakan.

```
com.amazonaws.services.autoscaling.*
com.amazonaws.services.cloudformation.*
com.amazonaws.services.ec2.*
com.amazonaws.services.identitymanagement.*
```

### 3. Tentukan dependensi Maven v2 dari pernyataan impor v1

Nama layanan untuk v1 yang kami isolasi dari Langkah 2—misalnya `autoscaling` dan `cloudformation`—dapat dipetakan ke nama layanan v2 yang sama untuk sebagian besar. Karena artifactID Maven v2 cocok dengan nama layanan dalam banyak kasus, Anda memiliki informasi yang Anda butuhkan untuk menambahkan blok ketergantungan ke file POM Anda.

Tabel berikut menunjukkan bagaimana kita dapat menentukan dependensi v2.

peta nama layanan v1 ke... nama paket	Peta nama layanan v2 ke... nama paket	v2 Ketergantungan Maven
ec2  com.amazonaws.services. <b>ec2</b> .*	ec2  software.amazon.awssdk.services. <b>ec2</b> .*	<pre>&lt;dependency&gt;   &lt;groupId&gt;software. amazon.awssdk&lt;/gro upId&gt;   &lt;artifactId&gt; <b>ec2</b>&lt;/ artifactId&gt; &lt;/dependency&gt;</pre>
penskalaan otomatis  com.amazonaws.services. <b>autoscaling</b> .*	penskalaan otomatis  software.amazon.awssdk.services. <b>autoscaling</b> .*	<pre>&lt;dependency&gt;   &lt;groupId&gt;software. amazon.awssdk&lt;/gro upId&gt;   &lt;artifactId&gt; <b>autoscali ng</b> &lt;/artifactId&gt; &lt;/dependency&gt;</pre>
pembentukan awan  com.amazonaws.services. <b>cloudform ation</b> .*	pembentukan awan  software.amazon.awssdk. <b>cloudform ation</b> .*	<pre>&lt;dependency&gt;   &lt;groupId&gt;software. amazon.awssdk&lt;/gro upId&gt;   &lt;artifactId&gt; <b>cloudform ation</b> &lt;/artifactId&gt; &lt;/dependency&gt;</pre>
manajemen identitas*	iam*	<pre>&lt;dependency&gt;</pre>

peta nama layanan v1 ke...	Peta nama layanan v2 ke...	v2 Ketergantungan Maven
nama paket	nama paket	
<code>com.amazonaws.services.identitymanagement.*</code>	<code>software.amazon.awssdk.iam.*</code>	<pre>&lt;groupId&gt;software.amazon.awssdk&lt;/groupId&gt; &lt;artifactId&gt; iam&lt;/artifactId&gt; &lt;/dependency&gt;</pre>

\* The identitymanagement to iam mapping adalah pengecualian di mana nama layanan yang digunakan dalam nama paket berbeda antar versi.

#### 4. Tambahkan elemen ketergantungan v2 ke file POM

Pada langkah 3, kami menentukan empat blok ketergantungan yang perlu ditambahkan ke file POM. Kami tidak perlu menambahkan versi karena kami telah menentukan BOM di langkah 1. Setelah impor ditambahkan, file POM kami memiliki elemen ketergantungan berikut.

```
...
<dependencies>
  ...
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>autoscaling</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>iam</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>cloudformation</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>ec2</artifactId>
  </dependency>
  ...
</dependencies>
```

...

## 5. Dalam file Java, secara bertahap ubah kelas v1 ke kelas v2

Dalam metode yang kita migrasi, kita lihat

- Klien layanan EC2 dari `com.amazonaws.services.ec2.AmazonEC2Client`.
- Beberapa kelas model EC2 digunakan. Misalnya `DescribeInstancesRequest` dan `DescribeInstancesResult`.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesResult;
import com.amazonaws.services.ec2.model.Instance;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Reservation;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;
...
private static List<Instance> getRunningInstances(AmazonEC2Client ec2, List<String>
instanceIds)
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = new DescribeInstancesRequest()
            .withInstanceIds(instanceIds);
        DescribeInstancesResult result;
        do {
            // DescribeInstancesResponse is a paginated response, so use tokens with
multiple re
            result = ec2.describeInstances(request);
            request.setNextToken(result.getNextToken()); // Prepare request for next
page.
            for (final Reservation r : result.getReservations()) {
                for (final Instance instance : r.getInstances()) {
                    LOGGER.info("Examining instanceId: "+ instance.getInstanceId());
                    // if instance is in a running state, add it to runningInstances
list.
```

```
        if (RUNNING_STATES.contains(instance.getState().getName())) {
            runningInstances.add(instance);
        }
    }
} while (result.getNextToken() != null);
} catch (final AmazonEC2Exception exception) {
    // if instance isn't found, assume its terminated and continue.
    if (exception.getErrorCode().equals(NOT_FOUND_ERROR_CODE)) {
        LOGGER.info("Instance probably terminated; moving on.");
    } else {
        throw exception;
    }
}
return runningInstances;
}
...

```

Tujuan kami adalah mengganti semua impor v1 dengan impor v2. Kami melanjutkan satu kelas pada satu waktu.

a. Ganti pernyataan impor atau nama kelas

Kita melihat bahwa parameter pertama untuk `describeRunningInstances` metode ini adalah `AmazonEC2Client` instance v1. Lakukan salah satu dari cara berikut:

- Ganti impor untuk `com.amazonaws.services.ec2.AmazonEC2Client` dengan `software.amazon.awssdk.services.ec2.Ec2Client` dan ubah `AmazonEC2Client` ke `Ec2Client`.
- Ubah tipe parameter menjadi `Ec2Client` dan biarkan IDE meminta kami untuk impor yang benar. IDE kami akan meminta kami untuk mengimpor kelas v2 karena nama klien berbeda—`AmazonEC2Client` dan `Ec2Client`. Pendekatan ini tidak berfungsi jika nama kelas sama di kedua versi.

b. Ganti kelas model v1 dengan setara v2

Setelah perubahan ke `v2Ec2Client`, jika kita menggunakan IDE, kita melihat kesalahan kompilasi dalam pernyataan berikut.

```
result = ec2.describeInstances(request);
```

Kesalahan kompilasi dihasilkan dari penggunaan instance v1 `DescribeInstancesRequest` sebagai parameter ke `Ec2Client describeInstances` metode v2. Untuk memperbaikinya, buat pernyataan penggantian atau impor berikut.

menggantikan	dengan
<pre>import com.amazonaws.services.ec2.model.DescribeInstancesRequest</pre>	<pre>import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest</pre>

c. Ubah konstruktor v1 menjadi pembangun v2.

Kami masih melihat kesalahan kompilasi karena [tidak ada konstruktor di kelas v2](#). Untuk memperbaikinya, buat perubahan berikut.

perubahan	kepada
<pre>final DescribeInstancesRequest request = new DescribeInstancesRequest().withInstanceIds(instanceIdsCopy);</pre>	<pre>final DescribeInstancesRequest request = DescribeInstancesRequest.builder().instanceIds(instanceIdsCopy).build();</pre>

d. Ganti objek **\*Result** respons v1 dengan ekuivalen v2 **\*Response**

Perbedaan yang konsisten antara v1 dan v2 adalah bahwa semua [objek respons di v2 diakhiri dengan \\*Response alih-alih. \\*Result](#) Ganti impor v1 ke `DescribeInstancesResult` impor v2, `DescribeInstancesResponse`.

d. Buat perubahan API

Pernyataan berikut membutuhkan beberapa perubahan.

```
request.setNextToken(result.getNextToken());
```



Dalam v2, [metode setter](#) tidak menggunakan set atau dengan prefix. Metode getter yang diawali dengan get juga hilang di SDK for Java 2.x

Kelas model, seperti request instance, tidak dapat diubah di v2, jadi kita perlu membuat yang baru `DescribeInstancesRequest` dengan pembangun.

Dalam v2, pernyataan menjadi sebagai berikut.

```
request = DescribeInstancesRequest.builder()
    .nextToken(result.nextToken())
    .build();
```

d. Ulangi sampai metode dikompilasi dengan kelas v2

Lanjutkan dengan sisa kode. Ganti impor v1 dengan impor v2 dan perbaiki kesalahan kompilasi. Lihat [Referensi API v2 dan Referensi apa yang berbeda](#) sesuai kebutuhan.

Setelah kami memigrasikan metode tunggal ini, kami memiliki kode v2 berikut.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;

import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.Reservation;
...
private static List<Instance> getRunningInstances(Ec2Client ec2, List<String>
instanceIds) {
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(instanceIds)
            .build();
```

```

        DescribeInstancesResponse result;
        do {
            // DescribeInstancesResponse is a paginated response, so use tokens
with multiple re
            result = ec2.describeInstances(request);
            request = DescribeInstancesRequest.builder() // Prepare request for
next page.
                .nextToken(result.nextToken())
                .build();
            for (final Reservation r : result.reservations()) {
                for (final Instance instance : r.instances()) {
                    // if instance is in a running state, add it to
runningInstances list.
                    if (RUNNING_STATES.contains(instance.state().nameAsString())) {
                        runningInstances.add(instance);
                    }
                }
            }
        } while (result.nextToken() != null);
    } catch (final Ec2Exception exception) {
        // if instance isn't found, assume its terminated and continue.
        if (exception.awsErrorDetails().errorCode().equals(NOT_FOUND_ERROR_CODE)) {
            LOGGER.info("Instance probably terminated; moving on.");
        } else {
            throw exception;
        }
    }
    return runningInstances;
}
...

```

Karena kami memigrasikan satu metode dalam file Java dengan delapan metode, kami memiliki campuran impor v1 dan v2 saat kami mengerjakan file tersebut. Kami menambahkan enam pernyataan impor terakhir saat kami melakukan langkah-langkah.

Setelah kami memigrasikan semua kode, tidak akan ada lagi pernyataan impor v1.

## 6. Hapus dependensi v1 Maven dari impor POM dan v1 dari file

Setelah kami memigrasikan semua kode v1 dalam file, kami memiliki pernyataan impor SDK v2 berikut.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.regions.ServiceMetadata;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.InstanceStateName;
import software.amazon.awssdk.services.ec2.model.Reservation;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesRequest;
```

Setelah kita memigrasikan semua file dalam aplikasi kita, kita tidak lagi membutuhkan dependensi v1 dalam file POM kita. Hapus v1 BOM dari bagian `DependencyManagement`, jika menggunakan, dan semua blok ketergantungan v1.

## 7. Memfaktorkan ulang kode untuk menggunakan penyempurnaan API v2

Untuk cuplikan yang telah kami migrasi, kami dapat menggunakan paginator v2 secara opsional dan membiarkan SDK mengelola permintaan berbasis token untuk lebih banyak data.

Kita dapat mengganti seluruh do klausa dengan yang berikut ini.

```
DescribeInstancesIterable responses =
ec2.describeInstancesPaginator(request);

responses.reservations().stream()
    .forEach(reservation -> reservation.instances()
        .forEach(instance -> {
            if
(RUNNING_STATES.contains(instance.state().nameAsString())) {
                runningInstances.put(instance.instanceId(),
instance);
            }
        }));
```

## Apa yang berbeda antara AWS SDK for Java 1.x dan 2.x

Bagian ini menjelaskan perubahan utama yang harus diperhatikan saat mengonversi aplikasi dari menggunakan AWS SDK for Java versi 1.x ke versi 2.x.

## Perubahan nama Package

Perubahan nyata dari SDK for Java 1.x ke SDK for Java 2.x adalah perubahan nama paket. Nama Package dimulai dengan `software.amazon.awssdk` SDK 2.x, sedangkan SDK 1.x menggunakan `com.amazonaws`

Nama yang sama ini membedakan artefak Maven dari SDK 1.x ke SDK 2.x. Artefak Maven untuk SDK 2.x menggunakan `software.amazon.awssdk` GroupId, sedangkan SDK 1.x menggunakan `com.amazonaws` GroupId.

Ada beberapa kali ketika kode Anda memerlukan `com.amazonaws` ketergantungan untuk proyek yang sebaliknya hanya menggunakan artefak SDK 2.x. Salah satu contohnya adalah ketika Anda bekerja dengan sisi server AWS Lambda. Ini ditunjukkan di bagian [Siapkan proyek Apache Maven](#) sebelumnya dalam panduan ini.

### Note

Beberapa nama paket di SDK 1.x berisi `v2`. Penggunaan `v2` dalam kasus ini biasanya berarti bahwa kode dalam paket ditargetkan untuk bekerja dengan versi 2 dari layanan.

Karena nama paket lengkap dimulai dengan `com.amazonaws`, ini adalah komponen SDK 1.x. Contoh nama paket ini di SDK 1.x adalah:

- `com.amazonaws.services.dynamodbv2`
- `com.amazonaws.retry.v2`
- `com.amazonaws.services.apigatewayv2`
- `com.amazonaws.services.simpleemailv2`

## Menambahkan versi 2.x ke proyek Anda

Maven adalah cara yang disarankan untuk mengelola dependensi saat menggunakan 2.x. AWS SDK for Java Untuk menambahkan komponen versi 2.x ke project Anda, perbarui `pom.xml` file Anda dengan ketergantungan pada SDK.

### Example

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

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>bom</artifactId>
  <version>2.16.1</version>
  <type>pom</type>
  <scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>

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

Anda juga dapat [menggunakan versi 1.x dan 2.x side-by-side](#) saat memigrasikan proyek ke versi 2.x.

## PojoS yang tidak dapat diubah

Klien dan permintaan operasi dan objek respons sekarang tidak dapat diubah dan tidak dapat diubah setelah pembuatan. Untuk menggunakan kembali variabel permintaan atau respons, Anda harus membangun objek baru untuk menetapkan objek tersebut.

### Example memperbarui objek permintaan di 1.x

```
DescribeAlarmsRequest request = new DescribeAlarmsRequest();
DescribeAlarmsResult response = cw.describeAlarms(request);

request.setNextToken(response.getNextToken());
```

### Example memperbarui objek permintaan di 2.x

```
DescribeAlarmsRequest request = DescribeAlarmsRequest.builder().build();
DescribeAlarmsResponse response = cw.describeAlarms(request);

request = DescribeAlarmsRequest.builder()
    .nextToken(response.nextToken())
    .build();
```

## Metode setter dan getter

Di AWS SDK for Java 2.x, nama metode penyetel tidak menyertakan awalan `set` atau `with`. Misalnya, `*.withEndpoint()` sekarang `*.endpoint()`.

Nama metode getter tidak menggunakan `get` awalan.

Example menggunakan metode setter di 1.x

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
    .withRegion("us-east-1")
    .build();
```

Example menggunakan metode setter di 2.x

```
DynamoDbClient client = DynamoDbClient.builder()
    .region(Region.US_EAST_1)
    .build();
```

Example menggunakan metode pengambil di 1.x

```
String token = request.getNextToken();
```

Example menggunakan metode pengambil di 2.x

```
String token = request.nextToken();
```

## Nama kelas model

Nama kelas model yang mewakili respons layanan diakhiri dengan `Response` in v2 alih-alih `Result` yang digunakan v1.

Example nama kelas yang mewakili respons di v1

```
CreateApiKeyResult
AllocateAddressResult
```

Example nama kelas yang mewakili respons di v2

```
CreateApiKeyResponse
```

## AllocateAddressResponse

## Status migrasi perpustakaan dan utilitas

### SDK for Java library dan utilitas

Tabel berikut mencantumkan status migrasi pustaka dan utilitas untuk SDK for Java.

Nama versi 1.12.x	Nama versi 2.x	Sejak versi di 2.x
DynamoDBMapper	<a href="#">DynamoDbEnhancedClient</a>	2.12.0
Pelayan	<a href="#">Pelayan</a>	2.15.0
CloudFrontUrlSigner, CloudFrontCookieSigner	<a href="#">CloudFrontUtilities</a>	2.18.33
TransferManager	<a href="#">S3 TransferManager</a>	2.19.0
Klien Metadata EC2	<a href="#">Klien Metadata EC2</a>	2.19.29
Pengurai URI S3	<a href="#">Pengurai URI S3</a>	2.20.41
Pembangun Kebijakan IAM	<a href="#">Pembangun Kebijakan IAM</a>	2.20.126
Buffering Sisi Klien Amazon SQS	Batching Permintaan Otomatis	<a href="#">belum dirilis</a>
Pendengar Kemajuan	Pendengar Kemajuan	<a href="#">belum dirilis</a>

### Perpustakaan terkait

Tabel berikut mencantumkan pustaka yang dirilis secara terpisah tetapi bekerja dengan SDK for Java 2.x.

Nama yang digunakan dengan versi 2.x dari SDK for Java	Sejak versi
<a href="#">Klien Enkripsi Amazon S3</a>	3.0.0 1

Nama yang digunakan dengan versi 2.x dari SDK for Java	Sejak versi
<a href="#">AWS Klien Enkripsi Database untuk DynamoDB</a>	3.0.0.2

<sup>1</sup> Klien enkripsi untuk Amazon S3 tersedia dengan menggunakan ketergantungan Maven berikut.

```
<dependency>
  <groupId>software.amazon.encryption.s3</groupId>
  <artifactId>amazon-s3-encryption-client-java</artifactId>
  <version>3.x</version>
</dependency>
```

<sup>2</sup> AWS Database Encryption Client untuk DynamoDB tersedia dengan menggunakan dependensi Maven berikut.

```
<dependency>
  <groupId>software.amazon.cryptography</groupId>
  <artifactId>aws-database-encryption-sdk-dynamodb</artifactId>
  <version>3.x</version>
</dependency>
```

## Detail migrasi untuk perpustakaan dan utilitas

- [Manajer Transfer S3](#)
- [Utilitas metadata EC2](#)
- [CloudFrontprepenandatangan](#)
- [Penguraian URI S3](#)

## Perubahan klien

### Pembangun klien

Anda harus membuat semua klien menggunakan metode pembangun klien. Konstruktor tidak lagi tersedia.



## Example membuat klien di versi 1.x

```
AmazonDynamoDB ddbClient = AmazonDynamoDBClientBuilder.defaultClient();
AmazonDynamoDBClient ddbClient = new AmazonDynamoDBClient();
```

## Example membuat klien di versi 2.x

```
DynamoDbClient ddbClient = DynamoDbClient.create();
DynamoDbClient ddbClient = DynamoDbClient.builder().build();
```

## Nama kelas klien

Semua nama kelas klien sekarang sepenuhnya berselubung unta dan tidak lagi diawali oleh. Amazon Perubahan ini selaras dengan nama yang digunakan dalam. AWS CLI

### Example nama kelas di 1.x

```
AmazonDynamoDB
AWSACMPAAsyncClient
```

### Example nama kelas di 2.x

```
DynamoDbClient
AcmAsyncClient
```

## Perubahan nama kelas klien

1.x Klien	2.x Klien
<code>com.amazonaws.services.acmpca.AWSACMPAAsyncClient</code>	<code>software.amazon.awssdk.services.acm.AcmAsyncClient</code>
<code>com.amazonaws.services.acmpca.AWSACMPAClient</code>	<code>software.amazon.awssdk.services.acm.AcmClient</code>
<code>com.amazonaws.services.alexforbusiness.AmazonAlexaForBusinessAsyncClient</code>	<code>software.amazon.awssdk.services.alexforbusiness.AlexaForBusinessAsyncClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.alex aforbusiness.AmazonAlexaFor BusinessClient</code>	<code>software.amazon.awssdk.serv ices.alexaforbusiness.Alexa ForBusinessClient</code>
<code>com.amazonaws.services.apig ateway.AmazonApiGatewayAsyn cClient</code>	<code>software.amazon.awssdk.serv ices.apigateway.ApiGatewayA syncClient</code>
<code>com.amazonaws.services.apig ateway.AmazonApiGatewayClient</code>	<code>software.amazon.awssdk.serv ices.apigateway.ApiGatewayC lient</code>
<code>com.amazonaws.services.appl icationautoscaling.AWSAppli cationAutoScalingAsyncClient</code>	<code>software.amazon.awssdk.serv ices.applicationautoscaling .ApplicationAutoScalingAsyn cClient</code>
<code>com.amazonaws.services.appl icationautoscaling.AWSAppli cationAutoScalingClient</code>	<code>software.amazon.awssdk.serv ices.applicationautoscaling .ApplicationAutoScalingClient</code>
<code>com.amazonaws.services.appl icationdiscovery.AWSAppl icationDiscoveryAsyncClient</code>	<code>software.amazon.awssdk.serv ices.applicationdiscovery.A pplicationDiscoveryAsyncClient</code>
<code>com.amazonaws.services.appl icationdiscovery.AWSAppl icationDiscoveryClient</code>	<code>software.amazon.awssdk.serv ices.applicationdiscovery.A pplicationDiscoveryClient</code>
<code>com.amazonaws.services.app stream.AmazonAppStreamAsynC lient</code>	<code>software.amazon.awssdk.serv ices.appstream.AppStreamAsy ncClient</code>
<code>com.amazonaws.services.app stream.AmazonAppStreamClient</code>	<code>software.amazon.awssdk.serv ices.appstream.AppStreamClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.appsync.AWSAppSyncAsyncClient</code>	<code>software.amazon.awssdk.services.appsync.AppSyncAsyncClient</code>
<code>com.amazonaws.services.appsync.AWSAppSyncClient</code>	<code>software.amazon.awssdk.services.appsync.AppSyncClient</code>
<code>com.amazonaws.services.athena.AmazonAthenaAsyncClient</code>	<code>software.amazon.awssdk.services.athena.AthenaAsyncClient</code>
<code>com.amazonaws.services.athena.AmazonAthenaClient</code>	<code>software.amazon.awssdk.services.athena.AthenaClient</code>
<code>com.amazonaws.services.autoscaling.AmazonAutoScalingAsyncClient</code>	<code>software.amazon.awssdk.services.autoscaling.AutoScalingAsyncClient</code>
<code>com.amazonaws.services.autoscaling.AmazonAutoScalingClient</code>	<code>software.amazon.awssdk.services.autoscaling.AutoScalingClient</code>
<code>com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansAsyncClient</code>	<code>software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansAsyncClient</code>
<code>com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansClient</code>	<code>software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansClient</code>
<code>com.amazonaws.services.batch.AWSBatchAsyncClient</code>	<code>software.amazon.awssdk.services.batch.BatchAsyncClient</code>
<code>com.amazonaws.services.batch.AWSBatchClient</code>	<code>software.amazon.awssdk.services.batch.BatchClient</code>
<code>com.amazonaws.services.budgets.AWSBudgetsAsyncClient</code>	<code>software.amazon.awssdk.services.budgets.BudgetsAsyncClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.budgets.AWSBudgetsClient</code>	<code>software.amazon.awssdk.services.budgets.BudgetsClient</code>
<code>com.amazonaws.services.certificatemanager.AWSCertificateManagerAsyncClient</code>	<code>software.amazon.awssdk.services.acm.AcmAsyncClient</code>
<code>com.amazonaws.services.certificatemanager.AWSCertificateManagerClient</code>	<code>software.amazon.awssdk.services.acm.AcmClient</code>
<code>com.amazonaws.services.cloud9.AWSCloud9AsyncClient</code>	<code>software.amazon.awssdk.services.cloud9.Cloud9AsyncClient</code>
<code>com.amazonaws.services.cloud9.AWSCloud9Client</code>	<code>software.amazon.awssdk.services.cloud9.Cloud9Client</code>
<code>com.amazonaws.services.clouddirectory.AmazonCloudDirectoryAsyncClient</code>	<code>software.amazon.awssdk.services.clouddirectory.CloudDirectoryAsyncClient</code>
<code>com.amazonaws.services.clouddirectory.AmazonCloudDirectoryClient</code>	<code>software.amazon.awssdk.services.clouddirectory.CloudDirectoryClient</code>
<code>com.amazonaws.services.cloudformation.AmazonCloudFormationAsyncClient</code>	<code>software.amazon.awssdk.services.cloudformation.CloudFormationAsyncClient</code>
<code>com.amazonaws.services.cloudformation.AmazonCloudFormationClient</code>	<code>software.amazon.awssdk.services.cloudformation.CloudFormationClient</code>
<code>com.amazonaws.services.cloudfront.AmazonCloudFrontAsyncClient</code>	<code>software.amazon.awssdk.services.cloudfront.CloudFrontAsyncClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.cloudfront.AmazonCloudFrontClient</code>	<code>software.amazon.awssdk.services.cloudfront.CloudFrontClient</code>
<code>com.amazonaws.services.cloudhsm.AWSCloudHSMAsyncClient</code>	<code>software.amazon.awssdk.services.cloudhsm.CloudHsmAsyncClient</code>
<code>com.amazonaws.services.cloudhsm.AWSCloudHSMClient</code>	<code>software.amazon.awssdk.services.cloudhsm.CloudHsmClient</code>
<code>com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2AsyncClient</code>	<code>software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2AsyncClient</code>
<code>com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2Client</code>	<code>software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2Client</code>
<code>com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainAsyncClient</code>	<code>software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainAsyncClient</code>
<code>com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainClient</code>	<code>software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainClient</code>
<code>com.amazonaws.services.cloudsearchv2.AmazonCloudSearchAsyncClient</code>	<code>software.amazon.awssdk.services.cloudsearch.CloudSearchAsyncClient</code>
<code>com.amazonaws.services.cloudsearchv2.AmazonCloudSearchClient</code>	<code>software.amazon.awssdk.services.cloudsearch.CloudSearchClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.cloudtrail.AWSCloudTrailAsyncClient</code>	<code>software.amazon.awssdk.services.cloudtrail.CloudTrailAsyncClient</code>
<code>com.amazonaws.services.cloudtrail.AWSCloudTrailClient</code>	<code>software.amazon.awssdk.services.cloudtrail.CloudTrailClient</code>
<code>com.amazonaws.services.cloudwatch.AmazonCloudWatchAsyncClient</code>	<code>software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient</code>
<code>com.amazonaws.services.cloudwatch.AmazonCloudWatchClient</code>	<code>software.amazon.awssdk.services.cloudwatch.CloudWatchClient</code>
<code>com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsAsyncClient</code>	<code>software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsAsyncClient</code>
<code>com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClient</code>	<code>software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient</code>
<code>com.amazonaws.services.codebuild.AWSCodeBuildAsyncClient</code>	<code>software.amazon.awssdk.services.codebuild.CodeBuildAsyncClient</code>
<code>com.amazonaws.services.codebuild.AWSCodeBuildClient</code>	<code>software.amazon.awssdk.services.codebuild.CodeBuildClient</code>
<code>com.amazonaws.services.codecommit.AWSCodeCommitAsyncClient</code>	<code>software.amazon.awssdk.services.codecommit.CodeCommitAsyncClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.codecommit.AWSCodeCommitClient</code>	<code>software.amazon.awssdk.services.codecommit.CodeCommitClient</code>
<code>com.amazonaws.services.codedeploy.AmazonCodeDeployAsyncClient</code>	<code>software.amazon.awssdk.services.codedeploy.CodeDeployAsyncClient</code>
<code>com.amazonaws.services.codedeploy.AmazonCodeDeployClient</code>	<code>software.amazon.awssdk.services.codedeploy.CodeDeployClient</code>
<code>com.amazonaws.services.codepipeline.AWSCodePipelineAsyncClient</code>	<code>software.amazon.awssdk.services.codepipeline.CodePipelineAsyncClient</code>
<code>com.amazonaws.services.codepipeline.AWSCodePipelineClient</code>	<code>software.amazon.awssdk.services.codepipeline.CodePipelineClient</code>
<code>com.amazonaws.services.codestar.AWSCodeStarAsyncClient</code>	<code>software.amazon.awssdk.services.codestar.CodeStarAsyncClient</code>
<code>com.amazonaws.services.codestar.AWSCodeStarClient</code>	<code>software.amazon.awssdk.services.codestar.CodeStarClient</code>
<code>com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityAsyncClient</code>	<code>software.amazon.awssdk.services.cognitoidentity.CognitoIdentityAsyncClient</code>
<code>com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityClient</code>	<code>software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.cognitoidp.AWSCognitoIdentityProviderAsyncClient</code>	<code>software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderAsyncClient</code>
<code>com.amazonaws.services.cognitoidp.AWSCognitoIdentityProviderClient</code>	<code>software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient</code>
<code>com.amazonaws.services.cognitosync.AmazonCognitoSyncAsyncClient</code>	<code>software.amazon.awssdk.services.cognitosync.CognitoSyncAsyncClient</code>
<code>com.amazonaws.services.cognitosync.AmazonCognitoSyncClient</code>	<code>software.amazon.awssdk.services.cognitosync.CognitoSyncClient</code>
<code>com.amazonaws.services.comprehend.AmazonComprehendAsyncClient</code>	<code>software.amazon.awssdk.services.comprehend.ComprehendAsyncClient</code>
<code>com.amazonaws.services.comprehend.AmazonComprehendClient</code>	<code>software.amazon.awssdk.services.comprehend.ComprehendClient</code>
<code>com.amazonaws.services.config.AmazonConfigAsyncClient</code>	<code>software.amazon.awssdk.services.config.ConfigAsyncClient</code>
<code>com.amazonaws.services.config.AmazonConfigClient</code>	<code>software.amazon.awssdk.services.config.ConfigClient</code>
<code>com.amazonaws.services.connect.AmazonConnectAsyncClient</code>	<code>software.amazon.awssdk.services.connect.ConnectAsyncClient</code>
<code>com.amazonaws.services.connect.AmazonConnectClient</code>	<code>software.amazon.awssdk.services.connect.ConnectClient</code>



1.x Klien	2.x Klien
<code>com.amazonaws.services.costandusagereport.AWSCostAndUsageReportAsyncClient</code>	<code>software.amazon.awssdk.services.costandusagereport.CostAndUsageReportAsyncClient</code>
<code>com.amazonaws.services.costandusagereport.AWSCostAndUsageReportClient</code>	<code>software.amazon.awssdk.services.costandusagereport.CostAndUsageReportClient</code>
<code>com.amazonaws.services.costexplorer.AWSCostExplorerAsyncClient</code>	<code>software.amazon.awssdk.services.costexplorer.CostExplorerAsyncClient</code>
<code>com.amazonaws.services.costexplorer.AWSCostExplorerClient</code>	<code>software.amazon.awssdk.services.costexplorer.CostExplorerClient</code>
<code>com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceAsyncClient</code>	<code>software.amazon.awssdk.services.databasemigration.DatabaseMigrationAsyncClient</code>
<code>com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceClient</code>	<code>software.amazon.awssdk.services.databasemigration.DatabaseMigrationClient</code>
<code>com.amazonaws.services.datapipeline.DataPipelineAsyncClient</code>	<code>software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient</code>
<code>com.amazonaws.services.datapipeline.DataPipelineClient</code>	<code>software.amazon.awssdk.services.datapipeline.DataPipelineClient</code>
<code>com.amazonaws.services.dax.AmazonDaxAsyncClient</code>	<code>software.amazon.awssdk.services.dax.DaxAsyncClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.dax. AmazonDaxClient</code>	<code>software.amazon.awssdk.serv ices.dax.DaxClient</code>
<code>com.amazonaws.services.devi cefarm.AWSDeviceFarmAsyncClient</code>	<code>software.amazon.awssdk.serv ices.devicefarm.DeviceFarmA syncClient</code>
<code>com.amazonaws.services.devi cefarm.AWSDeviceFarmClient</code>	<code>software.amazon.awssdk.serv ices.devicefarm.DeviceFarmC lient</code>
<code>com.amazonaws.services.dire ctconnect.AmazonDirectConne ctAsyncClient</code>	<code>software.amazon.awssdk.serv ices.directconnect.DirectCo nnectAsyncClient</code>
<code>com.amazonaws.services.dire ctconnect.AmazonDirectConne ctClient</code>	<code>software.amazon.awssdk.serv ices.directconnect.DirectCo nnectClient</code>
<code>com.amazonaws.services.dire ctory.AWSDirectoryServiceAs yncClient</code>	<code>software.amazon.awssdk.serv ices.directory.DirectoryAsy ncClient</code>
<code>com.amazonaws.services.dire ctory.AWSDirectoryServiceClient</code>	<code>software.amazon.awssdk.serv ices.directory.DirectoryClient</code>
<code>com.amazonaws.services.dlm. AmazonDLMAsyncClient</code>	<code>software.amazon.awssdk.serv ices.dlm.DlmAsyncClient</code>
<code>com.amazonaws.services.dlm. AmazonDLMClient</code>	<code>software.amazon.awssdk.serv ices.dlm.DlmClient</code>
<code>com.amazonaws.services.dyna modbv2.AmazonDynamoDBAsynC lient</code>	<code>software.amazon.awssdk.serv ices.dynamodb.DynamoDbAsynC Client</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBClient</code>	<code>software.amazon.awssdk.services.dynamodb.DynamoDbClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsAsyncClient</code>	<code>software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsAsyncClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsClient</code>	<code>software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsClient</code>
<code>com.amazonaws.services.ec2.AmazonEC2AsyncClient</code>	<code>software.amazon.awssdk.services.ec2.Ec2AsyncClient</code>
<code>com.amazonaws.services.ec2.AmazonEC2Client</code>	<code>software.amazon.awssdk.services.ec2.Ec2Client</code>
<code>com.amazonaws.services.ecr.AmazonECRAsyncClient</code>	<code>software.amazon.awssdk.services.ecr.EcrAsyncClient</code>
<code>com.amazonaws.services.ecr.AmazonECRClient</code>	<code>software.amazon.awssdk.services.ecr.EcrClient</code>
<code>com.amazonaws.services.ecs.AmazonECSAsyncClient</code>	<code>software.amazon.awssdk.services.ecs.EcsAsyncClient</code>
<code>com.amazonaws.services.ecs.AmazonECSClient</code>	<code>software.amazon.awssdk.services.ecs.EcsClient</code>
<code>com.amazonaws.services.eks.AmazonEKSAsyncClient</code>	<code>software.amazon.awssdk.services.eks.EksAsyncClient</code>
<code>com.amazonaws.services.eks.AmazonEKSClient</code>	<code>software.amazon.awssdk.services.eks.EksClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.elasticache.AmazonElasticCacheAsyncClient</code>	<code>software.amazon.awssdk.services.elasticache.ElasticCacheAsyncClient</code>
<code>com.amazonaws.services.elasticache.AmazonElasticCacheClient</code>	<code>software.amazon.awssdk.services.elasticache.ElasticCacheClient</code>
<code>com.amazonaws.services.elasticbeanstalk.AWSElasticBeanstalkAsyncClient</code>	<code>software.amazon.awssdk.services.elasticbeanstalk.ElasticBeanstalkAsyncClient</code>
<code>com.amazonaws.services.elasticbeanstalk.AWSElasticBeanstalkClient</code>	<code>software.amazon.awssdk.services.elasticbeanstalk.ElasticBeanstalkClient</code>
<code>com.amazonaws.services.elasticfilesystem.AmazonElasticFileSystemAsyncClient</code>	<code>software.amazon.awssdk.services.efs.EfsAsyncClient</code>
<code>com.amazonaws.services.elasticfilesystem.AmazonElasticFileSystemClient</code>	<code>software.amazon.awssdk.services.efs.EfsClient</code>
<code>com.amazonaws.services.elasticloadbalancing.AmazonElasticLoadBalancingAsyncClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingAsyncClient</code>
<code>com.amazonaws.services.elasticloadbalancing.AmazonElasticLoadBalancingClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingClient</code>
<code>com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingV2AsyncClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2AsyncClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2Client</code>
<code>com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceAsyncClient</code>	<code>software.amazon.awssdk.services.emr.EmrAsyncClient</code>
<code>com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClient</code>	<code>software.amazon.awssdk.services.emr.EmrClient</code>
<code>com.amazonaws.services.elasticsearch.AWSElasticsearchAsyncClient</code>	<code>software.amazon.awssdk.services.elasticsearch.ElasticsearchAsyncClient</code>
<code>com.amazonaws.services.elasticsearch.AWSElasticsearchClient</code>	<code>software.amazon.awssdk.services.elasticsearch.ElasticsearchClient</code>
<code>com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderAsyncClient</code>	<code>software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderAsyncClient</code>
<code>com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderClient</code>	<code>software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderClient</code>
<code>com.amazonaws.services.fms.AWSFMSAsyncClient</code>	<code>software.amazon.awssdk.services.fms.FmsAsyncClient</code>
<code>com.amazonaws.services.fms.AWSFMSClient</code>	<code>software.amazon.awssdk.services.fms.FmsClient</code>
<code>com.amazonaws.services.gamelift.AmazonGameLiftAsyncClient</code>	<code>software.amazon.awssdk.services.gamelift.GameLiftAsyncClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.gamelift.AmazonGameLiftClient</code>	<code>software.amazon.awssdk.services.gamelift.GameLiftClient</code>
<code>com.amazonaws.services.glacier.AmazonGlacierAsyncClient</code>	<code>software.amazon.awssdk.services.glacier.GlacierAsyncClient</code>
<code>com.amazonaws.services.glacier.AmazonGlacierClient</code>	<code>software.amazon.awssdk.services.glacier.GlacierClient</code>
<code>com.amazonaws.services.glue.AWSGlueAsyncClient</code>	<code>software.amazon.awssdk.services.glue.GlueAsyncClient</code>
<code>com.amazonaws.services.glue.AWSGlueClient</code>	<code>software.amazon.awssdk.services.glue.GlueClient</code>
<code>com.amazonaws.services.greengrass.AWSGreengrassAsyncClient</code>	<code>software.amazon.awssdk.services.greengrass.GreengrassAsyncClient</code>
<code>com.amazonaws.services.greengrass.AWSGreengrassClient</code>	<code>software.amazon.awssdk.services.greengrass.GreengrassClient</code>
<code>com.amazonaws.services.guardduty.AmazonGuardDutyAsyncClient</code>	<code>software.amazon.awssdk.services.guardduty.GuardDutyAsyncClient</code>
<code>com.amazonaws.services.guardduty.AmazonGuardDutyClient</code>	<code>software.amazon.awssdk.services.guardduty.GuardDutyClient</code>
<code>com.amazonaws.services.health.AWSHealthAsyncClient</code>	<code>software.amazon.awssdk.services.health.HealthAsyncClient</code>
<code>com.amazonaws.services.health.AWSHealthClient</code>	<code>software.amazon.awssdk.services.health.HealthClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.identitymanagement.AmazonIdentityManagementAsyncClient</code>	<code>software.amazon.awssdk.services.iam.IamAsyncClient</code>
<code>com.amazonaws.services.identitymanagement.AmazonIdentityManagementClient</code>	<code>software.amazon.awssdk.services.iam.IamClient</code>
<code>com.amazonaws.services.importexport.AmazonImportExportAsyncClient</code>	<code>software.amazon.awssdk.services.importexport.ImportExportAsyncClient</code>
<code>com.amazonaws.services.importexport.AmazonImportExportClient</code>	<code>software.amazon.awssdk.services.importexport.ImportExportClient</code>
<code>com.amazonaws.services.inspector.AmazonInspectorAsyncClient</code>	<code>software.amazon.awssdk.services.inspector.InspectorAsyncClient</code>
<code>com.amazonaws.services.inspector.AmazonInspectorClient</code>	<code>software.amazon.awssdk.services.inspector.InspectorClient</code>
<code>com.amazonaws.services.iot.AWSIoTAsyncClient</code>	<code>software.amazon.awssdk.services.iot.IotAsyncClient</code>
<code>com.amazonaws.services.iot.AWSIoTClient</code>	<code>software.amazon.awssdk.services.iot.IotClient</code>
<code>com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesAsyncClient</code>	<code>software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesAsyncClient</code>
<code>com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesClient</code>	<code>software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsAsyncClient</code>	<code>software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsAsyncClient</code>
<code>com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsClient</code>	<code>software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsClient</code>
<code>com.amazonaws.services.iotanalytics.AWSIoTAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.iotanalytics.IotAnalyticsAsyncClient</code>
<code>com.amazonaws.services.iotanalytics.AWSIoTAnalyticsClient</code>	<code>software.amazon.awssdk.services.iotanalytics.IotAnalyticsClient</code>
<code>com.amazonaws.services.iotdata.AWSIoTDataAsyncClient</code>	<code>software.amazon.awssdk.services.iotdata.IotDataAsyncClient</code>
<code>com.amazonaws.services.iotdata.AWSIoTDataClient</code>	<code>software.amazon.awssdk.services.iotdata.IotDataClient</code>
<code>com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneAsyncClient</code>	<code>software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient</code>
<code>com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneClient</code>	<code>software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient</code>
<code>com.amazonaws.services.kinesis.AmazonKinesisAsyncClient</code>	<code>software.amazon.awssdk.services.kinesis.KinesisAsyncClient</code>
<code>com.amazonaws.services.kinesis.AmazonKinesisClient</code>	<code>software.amazon.awssdk.services.kinesis.KinesisClient</code>



1.x Klien	2.x Klien
<code>com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsAsyncClient</code>
<code>com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsClient</code>	<code>software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsClient</code>
<code>com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseAsyncClient</code>	<code>software.amazon.awssdk.services.firehose.FirehoseAsyncClient</code>
<code>com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseClient</code>	<code>software.amazon.awssdk.services.firehose.FirehoseClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaClient</code>	<code>software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideo.KinesisVideoAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoClient</code>	<code>software.amazon.awssdk.services.kinesisvideo.KinesisVideoClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaClient</code>	<code>software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoPutMediaClient</code>	Tidak Didukung
<code>com.amazonaws.services.kms.AWSKMSAsyncClient</code>	<code>software.amazon.awssdk.services.kms.KmsAsyncClient</code>
<code>com.amazonaws.services.kms.AWSKMSClient</code>	<code>software.amazon.awssdk.services.kms.KmsClient</code>
<code>com.amazonaws.services.lambda.AWSLambdaAsyncClient</code>	<code>software.amazon.awssdk.services.lambda.LambdaAsyncClient</code>
<code>com.amazonaws.services.lambda.AWSLambdaClient</code>	<code>software.amazon.awssdk.services.lambda.LambdaClient</code>
<code>com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingAsyncClient</code>	<code>software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingAsyncClient</code>
<code>com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingClient</code>	<code>software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingClient</code>
<code>com.amazonaws.services.lexruntime.AmazonLexRuntimeAsyncClient</code>	<code>software.amazon.awssdk.services.lexruntime.LexRuntimeAsyncClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.lexruntime.AmazonLexRuntimeClient</code>	<code>software.amazon.awssdk.services.lexruntime.LexRuntimeClient</code>
<code>com.amazonaws.services.lightsail.AmazonLightsailAsyncClient</code>	<code>software.amazon.awssdk.services.lightsail.LightsailAsyncClient</code>
<code>com.amazonaws.services.lightsail.AmazonLightsailClient</code>	<code>software.amazon.awssdk.services.lightsail.LightsailClient</code>
<code>com.amazonaws.services.logs.AWSLogsAsyncClient</code>	<code>software.amazon.awssdk.services.logs.LogsAsyncClient</code>
<code>com.amazonaws.services.logs.AWSLogsClient</code>	<code>software.amazon.awssdk.services.logs.LogsClient</code>
<code>com.amazonaws.services.machinelearning.AmazonMachineLearningAsyncClient</code>	<code>software.amazon.awssdk.services.machinelearning.MachineLearningAsyncClient</code>
<code>com.amazonaws.services.machinelearning.AmazonMachineLearningClient</code>	<code>software.amazon.awssdk.services.machinelearning.MachineLearningClient</code>
<code>com.amazonaws.services.macie.AmazonMacieAsyncClient</code>	<code>software.amazon.awssdk.services.macie.MacieAsyncClient</code>
<code>com.amazonaws.services.macie.AmazonMacieClient</code>	<code>software.amazon.awssdk.services.macie.MacieClient</code>
<code>com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsAsyncClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsClient</code>	<code>software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsClient</code>
<code>com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementAsyncClient</code>	<code>software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementAsyncClient</code>
<code>com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementClient</code>	<code>software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementClient</code>
<code>com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringAsyncClient</code>	<code>software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringAsyncClient</code>
<code>com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringClient</code>	<code>software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringClient</code>
<code>com.amazonaws.services.mediaconvert.AWSMediaConvertAsyncClient</code>	<code>software.amazon.awssdk.services.mediaconvert.MediaConvertAsyncClient</code>
<code>com.amazonaws.services.mediaconvert.AWSMediaConvertClient</code>	<code>software.amazon.awssdk.services.mediaconvert.MediaConvertClient</code>
<code>com.amazonaws.services.medialive.AWSMediaLiveAsyncClient</code>	<code>software.amazon.awssdk.services.medialive.MediaLiveAsyncClient</code>
<code>com.amazonaws.services.medialive.AWSMediaLiveClient</code>	<code>software.amazon.awssdk.services.medialive.MediaLiveClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.mediapackage.AWSMediaPackageAsyncClient</code>	<code>software.amazon.awssdk.services.mediapackage.MediaPackageAsyncClient</code>
<code>com.amazonaws.services.mediapackage.AWSMediaPackageClient</code>	<code>software.amazon.awssdk.services.mediapackage.MediaPackageClient</code>
<code>com.amazonaws.services.mediastore.AWSMediaStoreAsyncClient</code>	<code>software.amazon.awssdk.services.mediastore.MediaStoreAsyncClient</code>
<code>com.amazonaws.services.mediastore.AWSMediaStoreClient</code>	<code>software.amazon.awssdk.services.mediastore.MediaStoreClient</code>
<code>com.amazonaws.services.mediastoredata.AWSMediaStoreDataAsyncClient</code>	<code>software.amazon.awssdk.services.mediastoredata.MediaStoreDataAsyncClient</code>
<code>com.amazonaws.services.mediastoredata.AWSMediaStoreDataClient</code>	<code>software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient</code>
<code>com.amazonaws.services.mediatailor.AWSMediaTailorAsyncClient</code>	<code>software.amazon.awssdk.services.mediatailor.MediaTailorAsyncClient</code>
<code>com.amazonaws.services.mediatailor.AWSMediaTailorClient</code>	<code>software.amazon.awssdk.services.mediatailor.MediaTailorClient</code>
<code>com.amazonaws.services.migrationhub.AWSMigrationHubAsyncClient</code>	<code>software.amazon.awssdk.services.migrationhub.MigrationHubAsyncClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.migrationhub.AWSMigrationHubClient</code>	<code>software.amazon.awssdk.services.migrationhub.MigrationHubClient</code>
<code>com.amazonaws.services.mobile.AWSMobileAsyncClient</code>	<code>software.amazon.awssdk.services.mobile.MobileAsyncClient</code>
<code>com.amazonaws.services.mobile.AWSMobileClient</code>	<code>software.amazon.awssdk.services.mobile.MobileClient</code>
<code>com.amazonaws.services.mq.AmazonMQAsyncClient</code>	<code>software.amazon.awssdk.services.mq.MqAsyncClient</code>
<code>com.amazonaws.services.mq.AmazonMQClient</code>	<code>software.amazon.awssdk.services.mq.MqClient</code>
<code>com.amazonaws.services.mturk.AmazonMTurkAsyncClient</code>	<code>software.amazon.awssdk.services.mturk.MTurkAsyncClient</code>
<code>com.amazonaws.services.mturk.AmazonMTurkClient</code>	<code>software.amazon.awssdk.services.mturk.MTurkClient</code>
<code>com.amazonaws.services.neptune.AmazonNeptuneAsyncClient</code>	<code>software.amazon.awssdk.services.neptune.NeptuneAsyncClient</code>
<code>com.amazonaws.services.neptune.AmazonNeptuneClient</code>	<code>software.amazon.awssdk.services.neptune.NeptuneClient</code>
<code>com.amazonaws.services.opsworks.AWSOpsWorksAsyncClient</code>	<code>software.amazon.awssdk.services.opsworks.OpsWorksAsyncClient</code>
<code>com.amazonaws.services.opsworks.AWSOpsWorksClient</code>	<code>software.amazon.awssdk.services.opsworks.OpsWorksClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.opsworkscom.AWSOpsWorksCMAsyncClient</code>	<code>software.amazon.awssdk.services.opsworkscom.OpsWorksCMAsyncClient</code>
<code>com.amazonaws.services.opsworkscom.AWSOpsWorksCMClient</code>	<code>software.amazon.awssdk.services.opsworkscom.OpsWorksCMClient</code>
<code>com.amazonaws.services.organizations.AWSOrganizationsAsyncClient</code>	<code>software.amazon.awssdk.services.organizations.OrganizationsAsyncClient</code>
<code>com.amazonaws.services.organizations.AWSOrganizationsClient</code>	<code>software.amazon.awssdk.services.organizations.OrganizationsClient</code>
<code>com.amazonaws.services.pi.AWSPIAsyncClient</code>	<code>software.amazon.awssdk.services.pi.PiAsyncClient</code>
<code>com.amazonaws.services.pi.AWSPIClient</code>	<code>software.amazon.awssdk.services.pi.PiClient</code>
<code>com.amazonaws.services.pinpoint.AmazonPinpointAsyncClient</code>	<code>software.amazon.awssdk.services.pinpoint.PinpointAsyncClient</code>
<code>com.amazonaws.services.pinpoint.AmazonPinpointClient</code>	<code>software.amazon.awssdk.services.pinpoint.PinpointClient</code>
<code>com.amazonaws.services.polly.AmazonPollyAsyncClient</code>	<code>software.amazon.awssdk.services.polly.PollyAsyncClient</code>
<code>com.amazonaws.services.polly.AmazonPollyClient</code>	<code>software.amazon.awssdk.services.polly.PollyClient</code>
<code>com.amazonaws.services.pricing.AWS PricingAsyncClient</code>	<code>software.amazon.awssdk.services.pricing.PricingAsyncClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.pricing.AWS PricingClient</code>	<code>software.amazon.awssdk.services.pricing.PricingClient</code>
<code>com.amazonaws.services.rds.AmazonRDSAsyncClient</code>	<code>software.amazon.awssdk.services.rds.RdsAsyncClient</code>
<code>com.amazonaws.services.rds.AmazonRDSClient</code>	<code>software.amazon.awssdk.services.rds.RdsClient</code>
<code>com.amazonaws.services.redshift.AmazonRedshiftAsyncClient</code>	<code>software.amazon.awssdk.services.redshift.RedshiftAsyncClient</code>
<code>com.amazonaws.services.redshift.AmazonRedshiftClient</code>	<code>software.amazon.awssdk.services.redshift.RedshiftClient</code>
<code>com.amazonaws.services.rekognition.AmazonRekognitionAsyncClient</code>	<code>software.amazon.awssdk.services.rekognition.RekognitionAsyncClient</code>
<code>com.amazonaws.services.rekognition.AmazonRekognitionClient</code>	<code>software.amazon.awssdk.services.rekognition.RekognitionClient</code>
<code>com.amazonaws.services.resourcegroups.AWSResourceGroupsAsyncClient</code>	<code>software.amazon.awssdk.services.resourcegroups.ResourceGroupsAsyncClient</code>
<code>com.amazonaws.services.resourcegroups.AWSResourceGroupsClient</code>	<code>software.amazon.awssdk.services.resourcegroups.ResourceGroupsClient</code>
<code>com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIAsyncClient</code>	<code>software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiAsyncClient</code>



1.x Klien	2.x Klien
<code>com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIClient</code>	<code>software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiClient</code>
<code>com.amazonaws.services.route53.AmazonRoute53AsyncClient</code>	<code>software.amazon.awssdk.services.route53.Route53AsyncClient</code>
<code>com.amazonaws.services.route53.AmazonRoute53Client</code>	<code>software.amazon.awssdk.services.route53.Route53Client</code>
<code>com.amazonaws.services.route53domains.AmazonRoute53DomainsAsyncClient</code>	<code>software.amazon.awssdk.services.route53domains.Route53DomainsAsyncClient</code>
<code>com.amazonaws.services.route53domains.AmazonRoute53DomainsClient</code>	<code>software.amazon.awssdk.services.route53domains.Route53DomainsClient</code>
<code>com.amazonaws.services.s3.AmazonS3Client</code>	<code>software.amazon.awssdk.services.s3.S3Client</code>
<code>com.amazonaws.services.sagemaker.AmazonSageMakerAsyncClient</code>	<code>software.amazon.awssdk.services.sagemaker.SageMakerAsyncClient</code>
<code>com.amazonaws.services.sagemaker.AmazonSageMakerClient</code>	<code>software.amazon.awssdk.services.sagemaker.SageMakerClient</code>
<code>com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeAsyncClient</code>	<code>software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeAsyncClient</code>
<code>com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeClient</code>	<code>software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.secretsmanager.AWSSecretsManagerAsyncClient</code>	<code>software.amazon.awssdk.services.secretsmanager.SecretsManagerAsyncClient</code>
<code>com.amazonaws.services.secretsmanager.AWSSecretsManagerClient</code>	<code>software.amazon.awssdk.services.secretsmanager.SecretsManagerClient</code>
<code>com.amazonaws.services.securitytoken.AWSSecurityTokenServiceAsyncClient</code>	<code>software.amazon.awssdk.services.sts.StsAsyncClient</code>
<code>com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClient</code>	<code>software.amazon.awssdk.services.sts.StsClient</code>
<code>com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryAsyncClient</code>	<code>software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryAsyncClient</code>
<code>com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryClient</code>	<code>software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryClient</code>
<code>com.amazonaws.services.servermigration.AWSServerMigrationAsyncClient</code>	<code>software.amazon.awssdk.services.sms.SmsAsyncClient</code>
<code>com.amazonaws.services.servermigration.AWSServerMigrationClient</code>	<code>software.amazon.awssdk.services.sms.SmsClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.servicecatalog.AWSServiceCatalogAsyncClient</code>	<code>software.amazon.awssdk.services.servicecatalog.ServiceCatalogAsyncClient</code>
<code>com.amazonaws.services.servicecatalog.AWSServiceCatalogClient</code>	<code>software.amazon.awssdk.services.servicecatalog.ServiceCatalogClient</code>
<code>com.amazonaws.services.servicediscovery.AWSServiceDiscoveryAsyncClient</code>	<code>software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryAsyncClient</code>
<code>com.amazonaws.services.servicediscovery.AWSServiceDiscoveryClient</code>	<code>software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryClient</code>
<code>com.amazonaws.services.shield.AWSShieldAsyncClient</code>	<code>software.amazon.awssdk.services.shield.ShieldAsyncClient</code>
<code>com.amazonaws.services.shield.AWSShieldClient</code>	<code>software.amazon.awssdk.services.shield.ShieldClient</code>
<code>com.amazonaws.services.simplesdb.AmazonSimpleDBAsyncClient</code>	<code>software.amazon.awssdk.services.simplesdb.SimpleDbAsyncClient</code>
<code>com.amazonaws.services.simplesdb.AmazonSimpleDBClient</code>	<code>software.amazon.awssdk.services.simplesdb.SimpleDbClient</code>
<code>com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceAsyncClient</code>	<code>software.amazon.awssdk.services.ses.SesAsyncClient</code>
<code>com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClient</code>	<code>software.amazon.awssdk.services.ses.SesClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.simp lesystemsmanagement.AWSSimp leSystemsManagementAsyncClient</code>	<code>software.amazon.awssdk.serv ices.ssm.SsmAsyncClient</code>
<code>com.amazonaws.services.simp lesystemsmanagement.AWSSimp leSystemsManagementClient</code>	<code>software.amazon.awssdk.serv ices.ssm.SsmClient</code>
<code>com.amazonaws.services.simp leworkflow.AmazonSimpleWork flowAsyncClient</code>	<code>software.amazon.awssdk.serv ices.swf.SwfAsyncClient</code>
<code>com.amazonaws.services.simp leworkflow.AmazonSimpleWork flowClient</code>	<code>software.amazon.awssdk.serv ices.swf.SwfClient</code>
<code>com.amazonaws.services.snow ball.AmazonSnowballAsyncClient</code>	<code>software.amazon.awssdk.serv ices.snowball.SnowballAsyn cClient</code>
<code>com.amazonaws.services.snow ball.AmazonSnowballClient</code>	<code>software.amazon.awssdk.serv ices.snowball.SnowballClient</code>
<code>com.amazonaws.services.sns. AmazonSNSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.sns.SnsAsyncClient</code>
<code>com.amazonaws.services.sns. AmazonSNSClient</code>	<code>software.amazon.awssdk.serv ices.sns.SnsClient</code>
<code>com.amazonaws.services.sqs. AmazonSQSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.sqs.SqsAsyncClient</code>
<code>com.amazonaws.services.sqs. AmazonSQSClient</code>	<code>software.amazon.awssdk.serv ices.sqs.SqsClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.stepfunctions.AWSStepFunctionsAsyncClient</code>	<code>software.amazon.awssdk.services.sfn.SfnAsyncClient</code>
<code>com.amazonaws.services.stepfunctions.AWSStepFunctionsClient</code>	<code>software.amazon.awssdk.services.sfn.SfnClient</code>
<code>com.amazonaws.services.storagegateway.AWSStorageGatewayAsyncClient</code>	<code>software.amazon.awssdk.services.storagegateway.StorageGatewayAsyncClient</code>
<code>com.amazonaws.services.storagegateway.AWSStorageGatewayClient</code>	<code>software.amazon.awssdk.services.storagegateway.StorageGatewayClient</code>
<code>com.amazonaws.services.support.AWSSupportAsyncClient</code>	<code>software.amazon.awssdk.services.support.SupportAsyncClient</code>
<code>com.amazonaws.services.support.AWSSupportClient</code>	<code>software.amazon.awssdk.services.support.SupportClient</code>
<code>com.amazonaws.services.transcribe.AmazonTranscribeAsyncClient</code>	<code>software.amazon.awssdk.services.transcribe.TranscribeAsyncClient</code>
<code>com.amazonaws.services.transcribe.AmazonTranscribeClient</code>	<code>software.amazon.awssdk.services.transcribe.TranscribeClient</code>
<code>com.amazonaws.services.translate.AmazonTranslateAsyncClient</code>	<code>software.amazon.awssdk.services.translate.TranslateAsyncClient</code>
<code>com.amazonaws.services.translate.AmazonTranslateClient</code>	<code>software.amazon.awssdk.services.translate.TranslateClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.waf.AWSWAFAsyncClient</code>	<code>software.amazon.awssdk.services.waf.WafAsyncClient</code>
<code>com.amazonaws.services.waf.AWSWAFClient</code>	<code>software.amazon.awssdk.services.waf.WafClient</code>
<code>com.amazonaws.services.waf.AWSWAFRegionalAsyncClient</code>	<code>software.amazon.awssdk.services.waf.regional.WafRegionalAsyncClient</code>
<code>com.amazonaws.services.waf.AWSWAFRegionalClient</code>	<code>software.amazon.awssdk.services.waf.regional.WafRegionalClient</code>
<code>com.amazonaws.services.workdocs.AmazonWorkDocsAsyncClient</code>	<code>software.amazon.awssdk.services.workdocs.WorkDocsAsyncClient</code>
<code>com.amazonaws.services.workdocs.AmazonWorkDocsClient</code>	<code>software.amazon.awssdk.services.workdocs.WorkDocsClient</code>
<code>com.amazonaws.services.workmail.AmazonWorkMailAsyncClient</code>	<code>software.amazon.awssdk.services.workmail.WorkMailAsyncClient</code>
<code>com.amazonaws.services.workmail.AmazonWorkMailClient</code>	<code>software.amazon.awssdk.services.workmail.WorkMailClient</code>
<code>com.amazonaws.services.workspaces.AmazonWorkSpacesAsyncClient</code>	<code>software.amazon.awssdk.services.workspaces.WorkSpacesAsyncClient</code>
<code>com.amazonaws.services.workspaces.AmazonWorkSpacesClient</code>	<code>software.amazon.awssdk.services.workspaces.WorkSpacesClient</code>

1.x Klien	2.x Klien
<code>com.amazonaws.services.xray.AWSXRayAsyncClient</code>	<code>software.amazon.awssdk.services.xray.XRayAsyncClient</code>
<code>com.amazonaws.services.xray.AWSXRayClient</code>	<code>software.amazon.awssdk.services.xray.XRayClient</code>

## Default pembuatan klien

Dalam versi 2.x, perubahan berikut telah dilakukan pada logika pembuatan klien default.

- Rantai penyedia kredensi default untuk S3 tidak lagi menyertakan kredensial anonim. Anda harus secara manual menentukan akses anonim ke S3 dengan menggunakan file. `AnonymousCredentialsProvider`
- Variabel lingkungan berikut yang terkait dengan pembuatan klien default berbeda.

1.x	2.x
<code>AWS_CBOR_DISABLED</code>	<code>CBOR_ENABLED</code>
<code>AWS_ION_BINARY_DISABLE</code>	<code>BINARY_ION_ENABLED</code>

- Properti sistem berikut yang terkait dengan pembuatan klien default berbeda.

1.x	2.x
<code>com.amazonaws.sdk.disableEc2Metadata</code>	<code>aws.disableEc2Metadata</code>
<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	<code>aws.ec2MetadataServiceEndpoint</code>
<code>com.amazonaws.sdk.disableCbor</code>	<code>aws.cborEnabled</code>
<code>com.amazonaws.sdk.disableIonBinary</code>	<code>aws.binaryIonEnabled</code>

- Versi 2.x tidak mendukung properti sistem berikut.

- 

1.x

```
com.amazonaws.sdk.disableCertChecking
```

```
com.amazonaws.sdk.enableDefaultMetrics
```

```
com.amazonaws.sdk.enableThrottledRetry
```

```
com.amazonaws.regions.RegionUtils.fileOverride
```

```
com.amazonaws.regions.RegionUtils.disableRemote
```

```
com.amazonaws.services.s3.disableImplicitGlobalClients
```

```
com.amazonaws.sdk.enableInRegionOptimizedMode
```

- Memuat konfigurasi Wilayah dari `endpoints.json` file kustom tidak lagi didukung.

## Konfigurasi klien

Di 1.x, konfigurasi klien SDK dimodifikasi dengan menyetel `ClientConfiguration` instance pada klien atau pembuat klien. Dalam versi 2.x, konfigurasi klien dibagi menjadi kelas konfigurasi terpisah. Dengan kelas konfigurasi terpisah, Anda dapat mengonfigurasi klien HTTP yang berbeda untuk klien async versus sinkron tetapi masih menggunakan kelas yang sama.

`ClientOverrideConfiguration`

Example konfigurasi klien dalam versi 1.x

```
AmazonDynamoDBClientBuilder.standard()  
.withClientConfiguration(clientConfiguration)  
.build()
```

Example konfigurasi klien sinkron di versi 2.x

```
ProxyConfiguration.Builder proxyConfig = ProxyConfiguration.builder();  
  
ApacheHttpClient.Builder httpClientBuilder =  
    ApacheHttpClient.builder()
```



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

ClientOverrideConfiguration.Builder overrideConfig =
    ClientOverrideConfiguration.builder();

DynamoDbClient client =
    DynamoDbClient.builder()
        .httpClientBuilder(httpClientBuilder)
        .overrideConfiguration(overrideConfig.build())
        .build();
```

## Example konfigurasi klien asinkron di versi 2.x

```
NettyNioAsyncHttpClient.Builder httpClientBuilder =
    NettyNioAsyncHttpClient.builder();

ClientOverrideConfiguration.Builder overrideConfig =
    ClientOverrideConfiguration.builder();

ClientAsyncConfiguration.Builder asyncConfig =
    ClientAsyncConfiguration.builder();

DynamoDbAsyncClient client =
    DynamoDbAsyncClient.builder()
        .httpClientBuilder(httpClientBuilder)
        .overrideConfiguration(overrideConfig.build())
        .asyncConfiguration(asyncConfig.build())
        .build();
```

## Klien HTTP

### Perubahan penting

- Di versi 2.x, Anda dapat mengubah klien HTTP mana yang akan digunakan saat runtime dengan menentukan implementasi yang digunakan. `clientBuilder.httpClientBuilder`
- Ketika Anda meneruskan klien HTTP dengan menggunakan `clientBuilder.httpClient` ke pembuat klien layanan, klien HTTP tidak ditutup secara default jika klien layanan ditutup. Ini memungkinkan Anda untuk berbagi klien HTTP antara klien layanan.
- Klien HTTP asinkron sekarang menggunakan IO non-pemblokiran.
- Beberapa operasi sekarang menggunakan HTTP/2 untuk meningkatkan kinerja.

## Pengaturan berubah

Pengaturan	1.x	2.x Sinkronisasi, Apache	2.x Asinkron, Netty
	<pre>ClientCon figuration clientConfig =     new ClientCon figuration()</pre>	<pre>ApacheHtt pClient.B uilder httpClien tBuilder =     ApacheHtt pClient.b uilder()</pre>	<pre>NettyNioA syncHttpC lient.Builder httpClien tBuilder =     NettyNioA syncHttpC lient.builder()</pre>
Koneksi maks	<pre>clientCon fig.setMa xConnecti ons(...) clientCon fig.withM axConnect ions(...)</pre>	<pre>httpClien tBuilder. maxConnec tions(...)</pre>	<pre>httpClien tBuilder. maxConcur rency(...)</pre>
Batas waktu koneksi	<pre>clientCon fig.setCo nnectionT imeout(...) clientConfig.wi thConnect ionTimeout(...)</pre>	<pre>httpClien tBuilder. connectio nTimeout(...)</pre>	<pre>httpClien tBuilder. connectio nTimeout(...)</pre>
Batas waktu soket	<pre>clientCon fig.setSo cketTimeo ut(...) clientConf ig.withSo cketTimeo ut(...)</pre>	<pre>httpClien tBuilder. socketTim eout(...)</pre>	<pre>httpClien tBuilder. writeTime out(...) httpClien tBuilder. readTimeo ut(...)</pre>

Pengaturan	1.x	2.x Sinkronisasi, Apache	2.x Asinkron, Netty
Koneksi TTL	<pre>clientConfig.setConnectionTTL(...) clientConfig.withConnectionTTL(...)</pre>	<pre>httpClientBuilder.connectionTimeToLive(...)</pre>	<pre>httpClientBuilder.connectionTimeToLive(...)</pre>
Koneksi maks idle	<pre>clientConfig.setConnectionMaxIdleMillis(...) clientConfig.withConnectionMaxIdleMillis(...)</pre>	<pre>httpClientBuilder.connectionMaxIdleTime(...)</pre>	<pre>httpClientBuilder.connectionMaxIdleTime(...)</pre>
Validasi setelah tidak aktif	<pre>clientConfig.setValidateAfterInactivityMillis(...) clientConfig.withValidateAfterInactivityMillis(...)</pre>	Tidak didukung ( <a href="#">Fitur Permintaan</a> )	Tidak didukung ( <a href="#">Fitur Permintaan</a> )

Pengaturan	1.x	2.x Sinkronisasi, Apache	2.x Asinkron, Netty
Alamat lokal	<pre>clientConfig.setLocalAddress(...) clientConfig.withLocalAddress(...)</pre>	<pre>httpClientBuilder.localAddress(...)</pre>	<a href="#">Tidak didukung</a>
Harapan-lanjutan diaktifkan	<pre>clientConfig.setUseExpectContinue(...) clientConfig.withUseExpectContinue(...)</pre>	<pre>httpClientBuilder.expectContinueEnabled(...)</pre>	Tidak didukung ( <a href="#">Fitur Permintaan</a> )
Penuai koneksi	<pre>clientConfig.setUseReaper(...) clientConfig.withReaper(...)</pre>	<pre>httpClientBuilder.useIdleConnectionReaper(...)</pre>	<pre>httpClientBuilder.useIdleConnectionReaper(...)</pre>
	<pre>AmazonDynamoDBClientBuilder     .standard()     .withClientConfiguration(clientConfiguration)     .build()</pre>	<pre>DynamoDBClient.builder()     .httpClientBuilder(httpClientBuilder)     .build()</pre>	<pre>DynamoDBAsyncClient.builder()     .httpClientBuilder(httpClientBuilder)     .build()</pre>

## Proksi klien HTTP

Pengaturan	1.x	2.x Sinkronisasi, Apache	2.x Asinkron, Netty
	<pre>ClientCon figuration clientConfig =     new ClientCon figuration()</pre>	<pre>ProxyConf iguration .Builder proxyConfig =     ProxyConf iguration .builder()</pre>	<pre>ProxyConf iguration .Builder proxyConfig =     ProxyConf iguration .builder()</pre>
Tuan rumah proxy	<pre>clientCon fig.setPr oxyHost(...) clientConfig.w ithProxyH ost(...)</pre>	<pre>proxyConf ig.endpoi nt(...)</pre>	<pre>proxyConf ig.host(...)</pre>
Port proxy	<pre>clientCon fig.setPr oxyPort(...) clientConfig.w ithProxyP ort(...)</pre>	<pre>proxyConf ig.endpoi nt(...)</pre> <p><a href="#">Port proxy</a> disematkan di endpoint</p>	<pre>proxyConf ig.port(...)</pre>
Nama pengguna proxy	<pre>clientCon fig.setPr oxyUserna me(...) clientConf ig.withPr oxyUserna me(...)</pre>	<pre>proxyConf ig.userna me(...)</pre>	<pre>proxyConf ig.userna me(...)</pre>

Pengaturan	1.x	2.x Sinkronisasi, Apache	2.x Asinkron, Netty
Kata sandi proxy	<pre>clientConfig.setProxyPassword(...) clientConfig.withProxyPassword(...)</pre>	<pre>proxyConfig.setPassword(...)</pre>	<pre>proxyConfig.setPassword(...)</pre>
Domain proxy	<pre>clientConfig.setProxyDomain(...) clientConfig.withProxyDomain(...)</pre>	<pre>proxyConfig.setNtlmDomain(...)</pre>	Tidak Didukung ( <a href="#">Fitur Permintaan</a> )
Stasiun kerja proxy	<pre>clientConfig.setProxyWorkspace(...) clientConfig.withProxyWorkstation(...)</pre>	<pre>proxyConfig.setNtlmWorkstation(...)</pre>	Tidak Didukung ( <a href="#">Fitur Permintaan</a> )
Metode otentikasi proxy	<pre>clientConfig.setProxyAuthenticationMethods(...) clientConfig.withProxyAuthenticationMethods(...)</pre>	<a href="#">Tidak Didukung</a>	Tidak Didukung ( <a href="#">Fitur Permintaan</a> )

Pengaturan	1.x	2.x Sinkronisasi, Apache	2.x Asinkron, Netty
Autentikasi proxy dasar preemptive	<pre>clientConfig.setPreemptiveBasicProxyAuth(...) clientConfig.withPreemptiveBasicProxyAuth(...)</pre>	<pre>proxyConfig.preemptiveBasicAuthenticationEnabled(...)</pre>	Tidak Didukung ( <a href="#">Fitur Permintaan</a> )
Host non-proxy	<pre>clientConfig.setNonProxyHosts(...) clientConfig.withNonProxyHosts(...)</pre>	<pre>proxyConfig.nonProxyHosts(...)</pre>	<pre>proxyConfig.nonProxyHosts(...)</pre>
Nonaktifkan proxy soket	<pre>clientConfig.setDisableSocketProxy(...) clientConfig.withDisableSocketProxy(...)</pre>	Tidak Didukung ( <a href="#">Fitur Permintaan</a> )	Tidak Didukung ( <a href="#">Fitur Permintaan</a> )

Pengaturan	1.x	2.x Sinkronisasi, Apache	2.x Asinkron, Netty
	<pre>AmazonDynamoDBClientBuilder .standard() .withClientConfiguration( clientConfiguration) .build()</pre>	<pre>httpClientBuilder.proxyConfiguration( proxyConfiguration).build()</pre>	<pre>httpClientBuilder.proxyConfiguration( proxyConfiguration).build()</pre>

### Pengesampingan klien

Pengaturan	1.x	2.x
	<pre>ClientConfiguration clientConfig = new ClientConfiguration()</pre>	<pre>ClientOverrideConfiguration.Builder overrideConfig = ClientOverrideConfiguration.builder()</pre>
Awalan agen pengguna	<pre>clientConfig.setUserAgentPrefix(...) clientConfig.withUserAgentPrefix(...)</pre>	<pre>overrideConfig.advancedOption( SdkAdvancedClientOption.USER_AGENT_PREFIX, ...)</pre>
Akhiran agen pengguna	<pre>clientConfig.setUserAgentSuffix(...) clientConfig.withUserAgentSuffix(...)</pre>	<pre>overrideConfig.advancedOption( SdkAdvancedClientOption.USER_AGENT_SUFFIX, ...)</pre>



Pengaturan	1.x	2.x
Signer	<pre>clientConfig.setSignerOverride(...) clientConfig.withSignerOverride(...)</pre>	<pre>overrideConfig.advancedOption(     SdkAdvancedClientOption.SIGNER, ...)</pre>
Header tambahan	<pre>clientConfig.addHeader(...) clientConfig.withHeader(...)</pre>	<pre>overrideConfig.putHeader(...)</pre>
Batas waktu permintaan	<pre>clientConfig.setRequestTimeout(...) clientConfig.withRequestTimeout(...)</pre>	<pre>overrideConfig.apiCallAttemptTimeout(...)</pre>
Batas waktu eksekusi klien	<pre>clientConfig.setClientExecutionTimeout(...) clientConfig.withClientExecutionTimeout(...)</pre>	<pre>overrideConfig.apiCallTimeout(...)</pre>
Gunakan Gzip	<pre>clientConfig.setUseGzip(...) clientConfig.withGzip(...)</pre>	Tidak Didukung ( <a href="#">Fitur Permintaan</a> )
Petunjuk ukuran penyangga soket	<pre>clientConfig.setSocketBufferSizeHints(...) clientConfig.withSocketBufferSizeHints(...)</pre>	Tidak Didukung ( <a href="#">Fitur Permintaan</a> )

Pengaturan	1.x	2.x
Metadata respons cache	<pre>clientConfig.setCacheResponseMetadata(...) clientConfig.withCacheResponseMetadata(...)</pre>	Tidak Didukung ( <a href="#">Fitur Permintaan</a> )
Ukuran cache metadata respons	<pre>clientConfig.setResponseMetadataCacheSize(...) clientConfig.withResponseMetadataCacheSize(...)</pre>	Tidak Didukung ( <a href="#">Fitur Permintaan</a> )
DNS resolver	<pre>clientConfig.setDnsResolver(...) clientConfig.withDnsResolver(...)</pre>	Tidak Didukung ( <a href="#">Fitur Permintaan</a> )
TCP keepalive	<pre>clientConfig.setUseTcpKeepAlive(...) clientConfig.withTcpKeepAlive(...)</pre>	<p>Opsi ini sekarang dalam konfigurasi HTTP Client</p> <pre>- ApacheHttpClient.builder().tcpKeepAlive(true) - NettyNioAsyncHttpClient.builder().tcpKeepAlive(true)</pre>
Amankan acak	<pre>clientConfig.setSecureRandom(...) clientConfig.withSecureRandom(...)</pre>	Tidak Didukung ( <a href="#">Fitur Permintaan</a> )

Pengaturan	1.x	2.x
	<pre>AmazonDynamoDBClientBuilder.standard()     .withClientConfiguration(clientConfiguration)     .build()</pre>	<pre>DynamoDbClient.builder()     .httpClientBuilder(httpClientBuilder)     .build()</pre>

## Klien mengganti percobaan ulang

Pengaturan	1.x	2.x
	<pre>ClientConfiguration clientConfig =     new ClientConfiguration()</pre>	<pre>RetryPolicy.Builder retryPolicy =     RetryPolicy.builder()</pre>
Kesalahan maks coba lagi	<pre>clientConfig.setMaxErrorRetry(...) clientConfig.withMaxErrorRetry(...)</pre>	<pre>retryPolicy.numRetries(...)</pre>
Gunakan percobaan ulang yang dibatasi	<pre>clientConfig.setUseThrottleRetries(...) clientConfig.withUseThrottleRetries(...)</pre>	<a href="#">Tidak didukung</a>
Maks percobaan ulang berturut-turut sebelum throttling	<pre>clientConfig.setMaxConsecutiveRetriesBeforeThrottling(...) clientConfig.withMaxConsecutiveRetriesBeforeThrottling(...)</pre>	<a href="#">Tidak didukung</a>

Pengaturan	1.x	2.x
	<pre>AmazonDynamoDBClientBuilder.standard()     .withClientConfiguration(clientConfiguration)     .build()</pre>	<pre>DynamoDbClient.builder()     .httpClientBuilder(httpClientBuilder)     .build()</pre>

## Klien asinkron

Pengaturan	1.x	2.x
		<pre>ClientAsyncConfiguration.Builder     asyncConfig =         ClientAsyncConfiguration.builder()</pre>
Pelaksana	<pre>AmazonDynamoDBAsyncClientBuilder.standard()     .withExecutorFactory(...)     .build()</pre>	<pre>asyncConfig.advancedOption(     SdkAdvancedAsyncClientOption.FUTURE_COMPLETION_EXECUTOR, ...)</pre>
		<pre>DynamoDbAsyncClient.builder()     .asyncConfiguration(asyncConfig)     .build()</pre>

## Perubahan klien lainnya

`ClientConfigurationOps` berikut dari 1.x telah berubah di 2.x SDK dan tidak memiliki ekuivalen langsung.

Pengaturan	1.x	2.x setara
Protokol	<pre>clientConfig.setProtocol(Protocol.HTTP) clientConfig.withProtocol(Protocol.HTTP)</pre>	<p>Pengaturan protokol adalah HTTPS secara default. Untuk mengubah pengaturan, tentukan protokol yang mengatur titik akhir HTTP pada pembuat klien:</p> <pre>clientBuilder.endpointOverride(     URI.create("http://..."))</pre>

## Perubahan penyedia kredensial

Bagian ini menyediakan pemetaan perubahan nama kelas penyedia kredensial dan metode antara versi 1.x dan 2.x dari AWS SDK for Java

### Perbedaan penting

- Penyedia kredensial default memuat properti sistem sebelum variabel lingkungan di versi 2.x. Untuk informasi selengkapnya, lihat [Menggunakan kredensial](#).
- Metode konstruktor diganti dengan builder metode create or.

#### Example

```
DefaultCredentialsProvider.create();
```

- Penyegaran asinkron tidak lagi disetel secara default. Anda harus menentukannya dengan builder penyedia kredensi.

#### Example

```
ContainerCredentialsProvider provider = ContainerCredentialsProvider.builder()
    .asyncCredentialUpdateEnabled(true)
    .build();
```

- Anda dapat menentukan jalur ke file profil khusus menggunakan `fileProfileCredentialsProvider.builder()`.

### Example

```
ProfileCredentialsProvider profile = ProfileCredentialsProvider.builder()
    .profileFile(ProfileFile.builder().content(Paths.get("myProfileFile.file")).build())
    .build();
```

- Format file profil telah berubah agar lebih cocok dengan file AWS CLI. Untuk detailnya, lihat [Mengonfigurasi AWS CLI](#) dalam Panduan AWS Command Line Interface Pengguna.

## Perubahan penyedia kredensial dipetakan antara versi 1.x dan 2.x

### AWSCredentialsProvider

Ubah kategori	1.x	2.x
Nama paket/kelas	<code>com.amazonaws.auth.AWSCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.AwsCredentialsProvider</code>
Nama metode	<code>getCredentials</code>	<code>resolveCredentials</code>
Metode yang tidak didukung	<code>refresh</code>	Tidak didukung

### DefaultAWSCredentialsProviderChain

Ubah kategori	1.x	2.x
Nama paket/kelas	<code>com.amazonaws.auth.DefaultAWSCredentialsProviderChain</code>	<code>software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider</code>

Ubah kategori	1.x	2.x
Pembuatan	<code>new DefaultAWSCredentialsProviderChain</code>	<code>DefaultCredentialsProvider.create</code>
Metode yang tidak didukung	<code>getInstance</code>	Tidak didukung
Urutan prioritas pengaturan eksternal	Variabel lingkungan sebelum properti sistem	Properti sistem sebelum variabel lingkungan

### **AWSStaticCredentialsProvider**

Ubah kategori	1.x	2.x
Nama paket/kelas	<code>com.amazonaws.auth.AWSStaticCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.StaticCredentialsProvider</code>
Pembuatan	<code>new AWSStaticCredentialsProvider</code>	<code>StaticCredentialsProvider.create</code>

### **EnvironmentVariableCredentialsProvider**

Ubah kategori	1.x	2.x
Nama paket/kelas	<code>com.amazonaws.auth.EnvironmentVariableCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider</code>
Pembuatan	<code>new EnvironmentVariableCredentialsProvider</code>	<code>EnvironmentVariableCredentialsProvider.create</code>

Ubah kategori	1.x	2.x
Nama variabel lingkungan	AWS_ACCESS_KEY	AWS_ACCESS_KEY_ID
	AWS_SECRET_KEY	AWS_SECRET_ACCESS_KEY

## SystemPropertiesCredentialsProvider

Ubah kategori	1.x	2.x
Nama paket/kelas	com.amazonaws.auth. SystemPropertiesC redentialsProvider	software.amazon.aw ssdk.auth.credenti als.SystemProperty CredentialsProvider
Pembuatan	new SystemPro pertiesCredentials Provider	SystemPropertiesCr edentialsProvider. create
Nama properti sistem	aws.secretKey	aws.secretAccessKey

## ProfileCredentialsProvider

Ubah kategori	1.x	2.x
Nama paket/kelas	com.amazonaws.auth .profile.ProfileCr edentialsProvider	software.amazon.aw ssdk.auth.credenti als.ProfileCredent ialsProvider
Pembuatan	new ProfileCr edentialsProvider	ProfileCredentials Provider.create
Lokasi profil kustom	<ul style="list-style-type: none"> <li>Variabel lingkungan AWS_CREDENTIAL_PRO FILES_FILE</li> </ul>	<ul style="list-style-type: none"> <li>Variabel lingkungan AWS_SHARED_CREDENT IALS_FILE</li> </ul>



Ubah kategori	1.x	2.x
	<ul style="list-style-type: none"> <li>• <code>new ProfileCredentialsProvider</code></li> </ul>	<ul style="list-style-type: none"> <li>• <code>ProfileCredentialsProvider.builder</code></li> </ul>

### ContainerCredentialsProvider

Ubah kategori	1.x	2.x
Nama paket/kelas	<code>com.amazonaws.auth.ContainerCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider</code>
Pembuatan	<code>new ContainerCredentialsProvider</code>	<code>ContainerCredentialsProvider.create</code>
Tentukan penyegaran asinkron	Tidak didukung	Perilaku default

### InstanceProfileCredentialsProvider

Ubah kategori	1.x	2.x
Nama paket/kelas	<code>com.amazonaws.auth.InstanceProfileCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider</code>
Pembuatan	<code>new InstanceProfileCredentialsProvider</code>	<code>InstanceProfileCredentialsProvider.create</code>
Tentukan penyegaran asinkron	<code>new InstanceProfileCredentialsProvider(true)</code>	<code>InstanceProfileCredentialProvider.builder().asyncCred</code>

Ubah kategori	1.x	2.x
		<code>essentialUpdateEnabled(true).build()</code>
Nama properti sistem	<code>com.amazonaws.sdk.disableEc2Metadata</code>	<code>aws.disableEc2Metadata</code>
	<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	<code>aws.ec2MetadataServiceEndpoint</code>

### STSAssumeRoleSessionCredentialsProvider

Ubah kategori	1.x	2.x
Nama paket/kelas	<code>com.amazonaws.auth.STSAssumeRoleSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsAssumeRoleCredentialsProvider</code>
Pembuatan	<ul style="list-style-type: none"> <li><code>new STSAssumeRoleSessionCredentialsProvider</code></li> <li><code>new STSAssumeRoleSessionCredentialsProvider.Builder</code></li> </ul>	<code>StsAssumeRoleCredentialsProvider.builder</code>
Penyegaran asinkron	Perilaku default	Perilaku default
Konfigurasi	<code>new STSAssumeRoleSessionCredentialsProvider.Builder</code>	Konfigurasi <code>AssumeRoleRequest</code> permintaan <code>StsClient</code> dan

**STSSessionCredentialsProvider**

Ubah kategori	1.x	2.x
Nama paket/kelas	<code>com.amazonaws.auth.STSSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsGetSessionTokenCredentialsProvider</code>
Pembuatan	<code>new STSAssumeRoleSessionCredentialsProvider</code>	<code>StsGetSessionTokenCredentialsProvider.builder</code>
Penyegaran asinkron	Perilaku default	<code>StsGetSessionTokenCredentialsProvider.builder</code>
Konfigurasi	Parameter konstruktor	Konfigurasi <code>getSessionTokenRequest</code> permintaan <code>StsClient</code> dan di pembangun

**WebIdentityFederationSessionCredentialsProvider**

Ubah kategori	1.x	2.x
Nama paket/kelas	<code>com.amazonaws.auth.WebIdentityFederationSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsAssumeRoleWithWebIdentityCredentialsProvider</code>
Pembuatan	<code>new WebIdentityFederationSessionCredentialsProvider</code>	<code>StsAssumeRoleWithWebIdentityCredentialsProvider.builder</code>

Ubah kategori	1.x	2.x
Penyegaran asinkron	Perilaku default	<code>StsAssumeRoleWithWebIdentityCredentialsProvider.builder</code>
Konfigurasi	Parameter konstruktor	Konfigurasi <code>AssumeRoleWithWebIdentityRequest</code> permintaan <code>StsClient</code> dan di bangun

### Kelas diganti

Kelas 1.x	2.x kelas pengganti
<code>com.amazonaws.auth.EC2ContainerCredentialsProviderWrapper</code>	<code>software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider</code> dan <code>software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider</code>
<code>com.amazonaws.services.s3.S3CredentialsProviderChain</code>	<code>software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider</code> dan <code>software.amazon.awssdk.auth.credentials.AnonymousCredentialsProvider</code>

### Kelas dihapus

Kelas 1.x
<code>com.amazonaws.auth.ClasspathPropertiesFileCredentialsProvider</code>

## Kelas 1.x

```
com.amazonaws.auth.PropertiesFileCredentialsProvider
```

## Perubahan wilayah

Bagian ini menjelaskan perubahan yang diterapkan di AWS SDK for Java 2.x untuk menggunakan `Region` dan `Regions` kelas.

### Konfigurasi wilayah

- Beberapa AWS layanan tidak memiliki titik akhir khusus Wilayah. Saat menggunakan layanan tersebut, Anda harus mengatur Wilayah sebagai `Region.AWS_GLOBAL` atau `Region.AWS_CN_GLOBAL`.

#### Example

```
Region region = Region.AWS_GLOBAL;
```

- `com.amazonaws.regions.Regions` dan `com.amazonaws.regions.Region` kelas sekarang digabungkan menjadi satu kelas, `software.amazon.awssdk.regions.Region`.

## Metode dan pemetaan nama kelas

Tabel berikut memetakan kelas terkait Wilayah antara versi 1.x dan 2.x dari AWS SDK for Java. Anda dapat membuat instance dari kelas-kelas ini menggunakan `of()` metode.

#### Example

```
RegionMetadata regionMetadata = RegionMetadata.of(Region.US_EAST_1);
```

### 1.x Perubahan metode kelas Wilayah

1.x	2.x
<code>Regions.fromName</code>	<code>Region.of</code>
<code>Regions.getName</code>	<code>Region.id</code>

1.x	2.x
<code>Regions.getDescription</code>	<code>Region.metadata().description()</code>
<code>Regions.getCurrentRegion</code>	Tidak Didukung
<code>Regions.DEFAULT_REGION</code>	Tidak Didukung
<code>Regions.name</code>	<code>Region.id</code>

### 1.x Perubahan metode kelas Wilayah

1.x	2.x
<code>Region.getName</code>	<code>Region.id</code>
<code>Region.hasHttpsEndpoint</code>	Tidak Didukung
<code>Region.hasHttpEndpoint</code>	Tidak Didukung
<code>Region.getAvailableEndpoints</code>	Tidak Didukung
<code>Region.createClient</code>	Tidak Didukung

### RegionMetadata perubahan metode kelas

1.x	2.x
<code>RegionMetadata.getName</code>	<code>RegionMetadata.name</code>
<code>RegionMetadata.getDomain</code>	<code>RegionMetadata.domain</code>
<code>RegionMetadata.getPartition</code>	<code>RegionMetadata.partition</code>

## ServiceMetadata perubahan metode kelas

1.x	2.x
<code>Region.getServiceEndpoint</code>	<code>ServiceMetadata.endpointFor(Region)</code>
<code>Region.isServiceSupported</code>	<code>ServiceMetadata.regions().contains(Region)</code>

## Operasi, permintaan, dan tanggapan berubah

Di v2.x SDK for Java, permintaan diteruskan ke operasi klien. Misalnya `DynamoDbClient`'s `PutItemRequest` diteruskan ke `DynamoDbClient.putItem` operasi. Operasi ini mengembalikan respons dari Layanan AWS, seperti `aPutItemResponse`.

Versi 2.x dari SDK for Java memiliki perubahan berikut dari 1.x.

- Operasi dengan beberapa halaman respons sekarang memiliki `Paginator` metode untuk iterasi secara otomatis atas semua item dalam respons.
- Anda tidak dapat mengubah permintaan dan tanggapan.
- Anda harus membuat permintaan dan tanggapan dengan metode pembangun statis alih-alih konstruktor. Misalnya, 1.x `new PutItemRequest().withTableName(...)` sekarang `PutItemRequest.builder().tableName(...).build()`.
- Operasi mendukung cara singkat untuk membuat permintaan: `dynamoDbClient.putItem(request -> request.tableName(...))`.

## Operasi streaming

Operasi streaming seperti Amazon S3 `getObject` dan `putObject` metode sekarang mendukung non-blocking I/O. Akibatnya, permintaan dan respons PoJOS tidak lagi mengambil sebagai parameter. `InputStream` Sebagai gantinya, untuk permintaan sinkron objek permintaan menerima `RequestBody`, yang merupakan aliran byte. Setara asinkron menerima sebuah `AsyncRequestBody`

## Example **putObject** operasi Amazon S3 di 1.x

```
s3client.putObject(BUCKET, KEY, new File(file_path));
```

## Example **putObject** operasi Amazon S3 di 2.x

```
s3client.putObject(PutObjectRequest.builder()
    .bucket(BUCKET)
    .key(KEY)
    .build(),
    RequestBody.of(Paths.get("myfile.in")));
```

Secara paralel, objek respons streaming menerima a `ResponseTransformer` untuk klien sinkron dan untuk klien asinkron. `AsyncResponseTransformer`

## Example **getObject** operasi Amazon S3 di 1.x

```
S3Object o = s3.getObject(bucket, key);
S3ObjectInputStream s3is = o.getObjectContent();
FileOutputStream fos = new FileOutputStream(new File(key));
```

## Example **getObject** operasi Amazon S3 di 2.x

```
s3client.getObject(GetObjectRequest.builder().bucket(bucket).key(key).build(),
    ResponseTransformer.toFile(Paths.get("key")));
```

Dalam SDK for Java 2.x, operasi respons streaming memiliki metode untuk memuat `AsBytes` respons ke dalam memori dan menyederhanakan konversi tipe dalam memori yang umum.

## Perubahan pengecualian

Nama kelas pengecualian, strukturnya, dan hubungannya telah berubah.

`software.amazon.awssdk.core.exception.SdkException` adalah `Exception` kelas dasar baru yang diperluas oleh semua pengecualian lainnya.

Tabel ini memetakan perubahan nama kelas pengecualian.



1.x	2.x
<code>com.amazonaws.SdkBaseException</code> <code>com.amazonaws.AmazonClientException</code>	<code>software.amazon.awssdk.core.exception.SdkException</code>
<code>com.amazonaws.SdkClientException</code>	<code>software.amazon.awssdk.core.exception.SdkClientException</code>
<code>com.amazonaws.AmazonServiceException</code>	<code>software.amazon.awssdk.awscore.exception.AwsServiceException</code>

Tabel berikut memetakan metode pada kelas pengecualian antara versi 1.x dan 2.x.

1.x	2.x
<code>AmazonServiceException.getRequestId</code>	<code>SdkServiceException.requestId</code>
<code>AmazonServiceException.getServiceName</code>	<code>AwsServiceException.awsErrorDetails().serviceName</code>
<code>AmazonServiceException.getErrorCode</code>	<code>AwsServiceException.awsErrorDetails().errorCode</code>
<code>AmazonServiceException.getErrorMessage</code>	<code>AwsServiceException.awsErrorDetails().errorMessage</code>
<code>AmazonServiceException.getStatusCode</code>	<code>AwsServiceException.awsErrorDetails().sdkHttpResponse().statusCode</code>
<code>AmazonServiceException.getHttpHeaders</code>	<code>AwsServiceException.awsErrorDetails().sdkHttpResponse().headers</code>

1.x	2.x
<code>AmazonServiceException.rawResponse</code>	<code>AwsServiceException.awsErrorResponseDetails().rawResponse</code>

## Perubahan serialisasi

SDK for Java v1.x dan v2.x berbeda dalam cara mereka membuat serial objek List untuk meminta parameter.

SDK for Java 1.x tidak membuat serial daftar kosong, sedangkan SDK for Java 2.x membuat serial daftar kosong sebagai parameter kosong.

Misalnya, pertimbangkan layanan dengan `SampleOperation` yang membutuhkan `aSampleRequest`. `SampleRequestMenerima` dua parameter—tipe String `str1` dan tipe List `listParam`—seperti yang ditunjukkan pada contoh berikut.

Example dari **SampleOperation** dalam 1.x

```
SampleRequest v1Request = new SampleRequest()
    .withStr1("TestName");

sampleServiceV1Client.sampleOperation(v1Request);
```

Pencatatan tingkat kabel menunjukkan bahwa `listParam` parameter tidak diserialisasi.

```
Action=SampleOperation&Version=2011-01-01&str1=TestName
```

Example dari **SampleOperation** dalam 2.x

```
sampleServiceV2Client.sampleOperation(b -> b
    .str1("TestName"));
```

Pencatatan tingkat kabel menunjukkan bahwa `listParam` parameter diserialkan tanpa nilai.

```
Action=SampleOperation&Version=2011-01-01&str1=TestName&listParam=
```

## Perubahan khusus layanan

### Amazon S3 berubah

SDK for Java 2.x menonaktifkan akses anonim secara default. Akibatnya, Anda harus mengaktifkan akses anonim dengan menggunakan `fileAnonymousCredentialsProvider`.

### Perubahan nama operasi

Banyak nama operasi untuk Amazon S3 klien telah berubah di AWS SDK for Java 2.x. Dalam versi 1.x, Amazon S3 klien tidak dihasilkan langsung dari API layanan. Hal ini mengakibatkan ketidakkonsistenan antara operasi SDK dan API layanan. Di versi 2.x, kami sekarang menghasilkan Amazon S3 klien agar lebih konsisten dengan API layanan.

Tabel berikut menunjukkan nama operasi dalam dua versi.

### Nama Operasi Amazon S3

1.x	2.x
<code>abortMultipartUpload</code>	<code>abortMultipartUpload</code>
<code>changeObjectStorageClass</code>	<code>copyObject</code>
<code>completeMultipartUpload</code>	<code>completeMultipartUpload</code>
<code>copyObject</code>	<code>copyObject</code>
<code>copyPart</code>	<code>uploadPartCopy</code>
<code>createBucket</code>	<code>createBucket</code>
<code>deleteBucket</code>	<code>deleteBucket</code>
<code>deleteBucketAnalyticsConfiguration</code>	<code>deleteBucketAnalyticsConfiguration</code>
<code>deleteBucketCrossOriginConfiguration</code>	<code>deleteBucketCors</code>
<code>deleteBucketEncryption</code>	<code>deleteBucketEncryption</code>

1.x	2.x
<code>deleteBucketInventoryConfiguration</code>	<code>deleteBucketInventoryConfiguration</code>
<code>deleteBucketLifecycleConfiguration</code>	<code>deleteBucketLifecycle</code>
<code>deleteBucketMetricsConfiguration</code>	<code>deleteBucketMetricsConfiguration</code>
<code>deleteBucketPolicy</code>	<code>deleteBucketPolicy</code>
<code>deleteBucketReplicationConfiguration</code>	<code>deleteBucketReplication</code>
<code>deleteBucketTaggingConfiguration</code>	<code>deleteBucketTagging</code>
<code>deleteBucketWebsiteConfiguration</code>	<code>deleteBucketWebsite</code>
<code>deleteObject</code>	<code>deleteObject</code>
<code>deleteObjectTagging</code>	<code>deleteObjectTagging</code>
<code>deleteObjects</code>	<code>deleteObjects</code>
<code>deleteVersion</code>	<code>deleteObject</code>
<code>disableRequesterPays</code>	<code>putBucketRequestPayment</code>
<code>doesBucketExist</code>	<code>headBucket</code>
<code>doesBucketExistV2</code>	<code>headBucket</code>
<code>doesObjectExist</code>	<code>headObject</code>
<code>enableRequesterPays</code>	<code>putBucketRequestPayment</code>
<code>generatePresignedUrl</code>	<a href="#">S3Presigner</a>
<code>getBucketAccelerateConfiguration</code>	<code>getBucketAccelerateConfiguration</code>

1.x	2.x
<code>getBucketAcl</code>	<code>getBucketAcl</code>
<code>getBucketAnalyticsConfiguration</code>	<code>getBucketAnalyticsConfiguration</code>
<code>getBucketCrossOriginConfiguration</code>	<code>getBucketCors</code>
<code>getBucketEncryption</code>	<code>getBucketEncryption</code>
<code>getBucketInventoryConfiguration</code>	<code>getBucketInventoryConfiguration</code>
<code>getBucketLifecycleConfiguration</code>	<code>getBucketLifecycle</code> atau <code>getBucketLifecycleConfiguration</code>
<code>getBucketLocation</code>	<code>getBucketLocation</code>
<code>getBucketLoggingConfiguration</code>	<code>getBucketLogging</code>
<code>getBucketMetricsConfiguration</code>	<code>getBucketMetricsConfiguration</code>
<code>getBucketNotificationConfiguration</code>	<code>getBucketNotification</code> atau <code>getBucketNotificationConfiguration</code>
<code>getBucketPolicy</code>	<code>getBucketPolicy</code>
<code>getBucketReplicationConfiguration</code>	<code>getBucketReplication</code>
<code>getBucketTaggingConfiguration</code>	<code>getBucketTagging</code>
<code>getBucketVersioningConfiguration</code>	<code>getBucketVersioning</code>
<code>getBucketWebsiteConfiguration</code>	<code>getBucketWebsite</code>
<code>getObject</code>	<code>getObject</code>
<code>getObjectAcl</code>	<code>getObjectAcl</code>

1.x	2.x
<code>getObjectAsString</code>	<code>getObjectAsBytes().asUtf8String</code>
<code>getObjectMetadata</code>	<code>headObject</code>
<code>getObjectTagging</code>	<code>getObjectTagging</code>
<code>getResourceUrl</code>	<a href="#">S3Utilities#getUrl</a>
<code>getS3AccountOwner</code>	<code>listBuckets</code>
<code>getUrl</code>	<a href="#">S3Utilities#getUrl</a>
<code>headBucket</code>	<code>headBucket</code>
<code>initiateMultipartUpload</code>	<code>createMultipartUpload</code>
<code>isRequesterPaysEnabled</code>	<code>getBucketRequestPayment</code>
<code>listBucketAnalyticsConfigurations</code>	<code>listBucketAnalyticsConfigurations</code>
<code>listBucketInventoryConfigurations</code>	<code>listBucketInventoryConfigurations</code>
<code>listBucketMetricsConfigurations</code>	<code>listBucketMetricsConfigurations</code>
<code>listBuckets</code>	<code>listBuckets</code>
<code>listMultipartUploads</code>	<code>listMultipartUploads</code>
<code>listNextBatchOfObjects</code>	<code>listObjectsV2Paginator</code>
<code>listNextBatchOfVersions</code>	<code>listObjectVersionsPaginator</code>
<code>listObjects</code>	<code>listObjects</code>
<code>listObjectsV2</code>	<code>listObjectsV2</code>
<code>listParts</code>	<code>listParts</code>

1.x	2.x
<code>listVersions</code>	<code>listObjectVersions</code>
<code>putObject</code>	<code>putObject</code>
<code>restoreObject</code>	<code>restoreObject</code>
<code>restoreObjectV2</code>	<code>restoreObject</code>
<code>selectObjectContent</code>	<code>selectObjectContent</code>
<code>setBucketAccelerateConfiguration</code>	<code>putBucketAccelerateConfiguration</code>
<code>setBucketAcl</code>	<code>putBucketAcl</code>
<code>setBucketAnalyticsConfiguration</code>	<code>putBucketAnalyticsConfiguration</code>
<code>setBucketCrossOriginConfiguration</code>	<code>putBucketCors</code>
<code>setBucketEncryption</code>	<code>putBucketEncryption</code>
<code>setBucketInventoryConfiguration</code>	<code>putBucketInventoryConfiguration</code>
<code>setBucketLifecycleConfiguration</code>	<code>putBucketLifecycle</code> atau <code>putBucketLifecycleConfiguration</code>
<code>setBucketLoggingConfiguration</code>	<code>putBucketLogging</code>
<code>setBucketMetricsConfiguration</code>	<code>putBucketMetricsConfiguration</code>
<code>setBucketNotificationConfiguration</code>	<code>putBucketNotification</code> atau <code>putBucketNotificationConfiguration</code>
<code>setBucketPolicy</code>	<code>putBucketPolicy</code>
<code>setBucketReplicationConfiguration</code>	<code>putBucketReplication</code>

1.x	2.x
<code>setBucketTaggingConfiguration</code>	<code>putBucketTagging</code>
<code>setBucketVersioningConfiguration</code>	<code>putBucketVersioning</code>
<code>setBucketWebsiteConfiguration</code>	<code>putBucketWebsite</code>
<code>setObjectAcl</code>	<code>putObjectAcl</code>
<code>setObjectRedirectLocation</code>	<code>copyObject</code>
<code>setObjectTagging</code>	<code>putObjectTagging</code>
<code>uploadPart</code>	<code>uploadPart</code>

## Amazon SNS berubah

Klien SNS tidak dapat lagi mengakses topik SNS di Wilayah selain Wilayah yang dikonfigurasi untuk diakses.

## Amazon SQS berubah

Klien SQS tidak dapat lagi mengakses antrian SQS di Wilayah selain Wilayah yang dikonfigurasi untuk diakses.

## Amazon RDS berubah

SDK for Java 2.x `RdsUtilities#generateAuthenticationToken` digunakan sebagai pengganti `RdsIamAuthTokenGenerator` kelas di 1.x.

## Perubahan file profil

AWS SDK for Java 2.x Mem-parsing definisi profil `~/.aws/config` dan `~/.aws/credentials` untuk lebih dekat meniru cara AWS CLI mem-parsing file.

SDK for Java 2.x:

- Menyelesaikan `~/` atau `~` diikuti oleh pemisah jalur default sistem file di awal jalur dengan memeriksa, secara berurutan,, `$USERPROFILE` (hanya Windows) `$HOME`, `$HOMEDRIVE`, `$HOMEPATH` (hanya Windows), dan kemudian properti sistem. `user.home`



- Mencari variabel `AWS_SHARED_CREDENTIALS_FILE` lingkungan alih-alih `AWS_CREDENTIAL_PROFILES_FILE`.
- Diam-diam menjatuhkan definisi profil dalam file konfigurasi tanpa kata `profile` di awal nama profil.
- Secara diam-diam menjatuhkan definisi profil yang tidak terdiri dari karakter alfanumerik, garis bawah atau tanda hubung (setelah `profile` kata utama dihapus untuk file konfigurasi).
- Menggabungkan pengaturan definisi profil yang digandakan dalam file yang sama.
- Menggabungkan pengaturan definisi profil yang digandakan dalam file konfigurasi dan kredensial.
- TIDAK menggabungkan pengaturan jika keduanya `[profile foo]` dan `[foo]` ditemukan dalam file yang sama.
- Menggunakan pengaturan di `[profile foo]` jika `[foo]` keduanya `[profile foo]` dan ditemukan di file konfigurasi.
- Menggunakan nilai pengaturan duplikat terakhir dalam file dan profil yang sama.
- Mengenali keduanya `;` dan `#` untuk mendefinisikan komentar.
- Mengenali `;` dan `#` dalam definisi profil untuk mendefinisikan komentar, bahkan jika karakter berdekatan dengan braket penutup.
- Mengenali `;` dan `#` mendefinisikan komentar hanya dalam menetapkan nilai hanya jika mereka didahului oleh spasi putih.
- Mengenali `;` dan `#` dan semua konten berikut dalam menetapkan nilai jika tidak didahului oleh spasi putih.
- Mempertimbangkan kredensial berbasis peran sebagai kredensial prioritas tertinggi. SDK 2.x selalu menggunakan kredensial berbasis peran jika pengguna menentukan properti. `role_arn`
- Mempertimbangkan kredensial berbasis sesi sebagai kredensial. `second-highest-priority` SDK 2.x selalu menggunakan kredensial berbasis sesi jika kredensial berbasis peran tidak digunakan dan pengguna menentukan properti dan. `aws_access_key_id` `aws_session_token`
- Menggunakan kredensial dasar jika kredensial berbasis peran dan sesi tidak digunakan dan pengguna menentukan properti. `aws_access_key_id`

## Variabel lingkungan dan properti sistem berubah

1.x Variabel Lingkungan	1.x Properti Sistem	2.x Variabel Lingkungan	2.x Properti Sistem
AWS_ACCESS_KEY_ID AWS_ACCESS_KEY	<code>aws.accessKeyId</code>	AWS_ACCESS_KEY_ID	<code>aws.accessKeyId</code>
AWS_SECRET_KEY AWS_SECRET_ACCESS_KEY	<code>aws.secretKey</code>	AWS_SECRET_ACCESS_KEY	<code>aws.secretAccessKey</code>
AWS_SESSION_TOKEN	<code>aws.sessionToken</code>	AWS_SESSION_TOKEN	<code>aws.sessionToken</code>
AWS_REGION	<code>aws.region</code>	AWS_REGION	<code>aws.region</code>
AWS_CONFIG_FILE		AWS_CONFIG_FILE	<code>aws.configFile</code>
AWS_CREDENTIALS_PROFILE_FILE		AWS_SHARED_CREDENTIALS_FILE	<code>aws.sharedCredentialsFile</code>
AWS_PROFILE	<code>aws.profile</code>	AWS_PROFILE	<code>aws.profile</code>
AWS_EC2_METADATA_DISABLED	<code>com.amazonaws.sdk.disableEc2Metadata</code>	AWS_EC2_METADATA_DISABLED	<code>aws.disableEc2Metadata</code>
	<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	AWS_EC2_METADATA_SERVICE_ENDPOINT	<code>aws.ec2MetadataServiceEndpoint</code>

1.x Variabel Lingkungan	1.x Properti Sistem	2.x Variabel Lingkungan	2.x Properti Sistem
AWS_CONTAINER_CREDENTIALS_RELATIVE_URI		AWS_CONTAINER_CREDENTIALS_RELATIVE_URI	<code>aws.containerCredentialsPath</code>
AWS_CONTAINER_CREDENTIALS_FULL_URI		AWS_CONTAINER_CREDENTIALS_FULL_URI	<code>aws.containerCredentialsFullUri</code>
AWS_CONTAINER_AUTHORIZATION_TOKEN		AWS_CONTAINER_AUTHORIZATION_TOKEN	<code>aws.containerAuthorizationToken</code>
AWS_CBOR_DISABLED	<code>com.amazonaws.sdk.disableCbor</code>	CBOR_ENABLED	<code>aws.cborEnabled</code>
AWS_ION_BINARY_DISABLE	<code>com.amazonaws.sdk.disableIonBinary</code>	BINARY_ION_ENABLED	<code>aws.binaryIonEnabled</code>
AWS_EXECUTION_ENV		AWS_EXECUTION_ENV	<code>aws.executionEnvironment</code>
	<code>com.amazonaws.sdk.disableCertificateChecking</code>	Tidak didukung ( <a href="#">fitur Permintaan</a> )	Tidak didukung ( <a href="#">fitur Permintaan</a> )

1.x Variabel Lingkungan	1.x Properti Sistem	2.x Variabel Lingkungan	2.x Properti Sistem
	<code>com.amazonaws.sdk.enableDefaultMetrics</code>	<a href="#">Tidak didukung</a>	<a href="#">Tidak didukung</a>
	<code>com.amazonaws.sdk.enableThrottledRetry</code>	<a href="#">Tidak didukung</a>	<a href="#">Tidak didukung</a>
	<code>com.amazonaws.regions.RegionUtils.fileOverride</code>	Tidak didukung ( <a href="#">fitur Permintaan</a> )	Tidak didukung ( <a href="#">fitur Permintaan</a> )
	<code>com.amazonaws.regions.RegionUtils.disableRemote</code>	Tidak didukung ( <a href="#">fitur Permintaan</a> )	Tidak didukung ( <a href="#">fitur Permintaan</a> )
	<code>com.amazonaws.services.s3.disableImplicitGlobalClients</code>	Tidak didukung ( <a href="#">fitur Permintaan</a> )	Tidak didukung ( <a href="#">fitur Permintaan</a> )
	<code>com.amazonaws.sdk.enableInRegionOptimizedMode</code>	Tidak didukung ( <a href="#">fitur Permintaan</a> )	Tidak didukung ( <a href="#">fitur Permintaan</a> )

## Perubahan Pelayan dari versi 1 ke versi 2

Topik ini merinci perubahan fungsionalitas Pelayan dari versi 1 (v1) ke versi 2 (v2).

Tabel berikut menunjukkan perbedaan untuk pelayan DynamoDB secara khusus. Pelayan untuk layanan lain mengikuti pola yang sama.

### Perubahan tingkat tinggi

Kelas pelayan berada dalam artefak Maven yang sama dengan layanan.

Perubahan	v1	v2
Ketergantungan Maven	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;       &lt;version&gt; 1.12.680<sup>1</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;     &lt;artifact Id&gt;dynamodb&lt;/artif actId&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; </pre>	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt;       &lt;version&gt; 2.25.10<sup>2</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;dynamodb&lt;/artif actId&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; </pre>

Perubahan	v1	v2
Nama paket	<code>com.amazonaws.services.dynamodbv2.waiters</code>	<code>software.amazon.awssdk.services.dynamodb.waiters</code>
Nama kelas	<a href="#">AmazonDynamoDBWaiters</a>	<ul style="list-style-type: none"> <li>Sinkron: <a href="#">DynamoDbWaiter</a></li> <li>Asinkron: <a href="#">DynamoDbAsyncWaiter</a></li> </ul>

<sup>1</sup> [Versi terbaru](#). <sup>2</sup> [Versi terbaru](#).

## Perubahan API

Perubahan	v1	v2
Buat pelayan	<pre>AmazonDynamoDB client = AmazonDynamoDBClientBuilder      .standard().build( ); AmazonDynamoDBWaiters waiter = client.waiters();</pre>	<p>Sinkron:</p> <pre>DynamoDbClient client = DynamoDbClient.create(); DynamoDbWaiter waiter = client.waiter();</pre> <p>Asinkron:</p> <pre>DynamoDbAsyncClient asyncClient =     DynamoDbAsyncClient.create(); DynamoDbAsyncWaiter waiter = asyncClient.waiter();</pre>
Tunggu sampai ada tabel	<p>Sinkron:</p> <pre>waiter.tableExists()</pre>	<p>Sinkron:</p>

Perubahan	v1	v2
	<pre data-bbox="609 212 1008 415"> .run(new WaiterParameters&lt;&gt;(     new DescribeTableRequest(tableName)))  Asinkron:  waiter.tableExists()     .runAsync(new WaiterParameters()         .withRequest(new DescribeTableRequest(tableName)),         new WaiterHandler() {             @Override             public void onSuccess(                 AmazonWebServiceRequest amazonWebServiceRequest) {                 System.out.println("Table creation succeeded");             }             @Override             public void onFailure(Exception e) {                 e.printStackTrace();             }         }).get(); </pre>	<pre data-bbox="1084 212 1500 716"> WaiterResponse&lt;DescribeTableResponse&gt; waiterResponse =     waiter.waitForTableExists(         r -&gt; r.tableName("myTable")); waiterResponse.matched().response()     .ifPresent(System.out::println);  Asinkron:  waiter.waitForTableExists(r -&gt;     r.tableName(tableName))     .whenComplete((r, t) -&gt; {         if (t != null) {             t.printStackTrace();         } else {             System.out.println("Table creation succeeded");         }     }).join(); </pre>

## Perubahan konfigurasi

Perubahan	v1	v2
Strategi Polling (upaya maksimal dan penundaan tetap)	<pre> MaxAttemptsRetryStrategy maxAttemptsRetryStrategy =     new MaxAttemptsRetryStrategy(10);  FixedDelayStrategy fixedDelayStrategy =     new FixedDelayStrategy(3);  PollingStrategy pollingStrategy =     new PollingStrategy(maxAttemptsRetryStrategy,         fixedDelayStrategy);  waiter.tableExists().run(     new WaiterParameters&lt;&gt;(         new DescribeTableRequest(tableName),         pollingStrategy); </pre>	<pre> FixedDelayBackoffStrategy fixedDelayBackoffStrategy =     FixedDelayBackoffStrategy.create(         Duration.ofSeconds(3));  waiter.waitUntilTableExists(r -&gt; r.tableName(tableName),     c -&gt; c.maxAttempts(10)         .backoffStrategy(fixedDelayBackoffStrategy)); </pre>

## Perubahan di Amazon S3 Transfer Manager dari versi 1 ke versi 2

Topik ini merinci perubahan di Amazon S3 Transfer Manager dari versi 1 (v1) ke versi 2 (v2).



## Perubahan tingkat tinggi

Perubahan	v1	v2
Ketergantungan Maven	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;       &lt;version&gt; 1.12.587<sup>1</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;     &lt;artifact Id&gt;aws-java-sdk-s3&lt;/ artifactId&gt;   &lt;/dependency&gt; &lt;/dependencies&gt; </pre>	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt;       &lt;version&gt; 2.21.21<sup>2</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifactId&gt;s3- transfer-manager&lt;/art ifactId&gt;   &lt;/dependency&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk.crt&lt;/groupId&gt;     &lt;artifact Id&gt;aws-crt&lt;/artifa ctId&gt;     &lt;version&gt; 0.28.7<sup>3</sup>&lt;/version&gt;   &lt;/dependency&gt; &lt;/dependencies&gt; </pre>

Perubahan	v1	v2
Nama paket	<code>com.amazonaws.serv ices.s3.transfer</code>	<code>software.amazon.aw ssdk.transfer.s3</code>
Nama kelas	<a href="#"><u>TransferManager</u></a>	<a href="#"><u>S3TransferManager</u></a>

<sup>1</sup> [Versi terbaru.](#) <sup>2</sup> [Versi terbaru.](#) <sup>3</sup> [Versi terbaru.](#)

## Perubahan API konfigurasi

Pengaturan	v1	v2
(dapatkan pembangun)	<pre>TransferManagerBuilder tmBuilder =     TransferManagerBui lder.standard();</pre>	<pre>S3TransferManager. Builder tmBuilder =     S3TransferManager. builder();</pre>
Klien S3	<pre>tmBuilder.withS3Cl ient(...); tmBuilder.setS3C lient(...);</pre>	<pre>tmBuilder.s3Client (...);</pre>
Pelaksana	<pre>tmBuilder.withExec utorFactory(...); tmBuilder.setExecu torFactory(...);</pre>	<pre>tmBuilder.executor (...);</pre>
Shutdown thread pool	<pre>tmBuilder.withShut DownThreadPools(...); tmBuilder.setS hutdownThreadPools (...);</pre>	Tidak didukung. Pelaksana yang disediakan tidak akan dimatikan saat S3 TransferManager ditutup
Ukuran bagian unggah minimum	<pre>tmBuilder.withMini mumUploadPartSize( ...);</pre>	<pre>S3AsyncClient s3 =     S3AsyncClient.crtB uilder().</pre>

Pengaturan	v1	v2
	<pre>tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>minimumPartSizeInBytes(...) .build();  tmBuilder.s3Client(s3);</pre>
Ambang batas unggahan multipart	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 =     S3AsyncClient.crtBuilder()         .thresholdInBytes(...)         .build();  tmBuilder.s3Client(s3);</pre>
Ukuran bagian salinan minimum	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 =     S3AsyncClient.crtBuilder()         .minimumPartSizeInBytes(...)         .build();  tmBuilder.s3Client(s3);</pre>
Ambang salinan multipart	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 =     S3AsyncClient.crtBuilder()         .thresholdInBytes(...)         .build();  tmBuilder.s3Client(s3);</pre>

Pengaturan	v1	v2
Nonaktifkan unduhan paralel	<pre>tmBuilder.withDisableParallelDownloads(...); tmBuilder.setDisableParallelDownloads(...);</pre>	<p>Nonaktifkan unduhan paralel dengan meneruskan klien S3 berbasis Java standar ke manajer transfer.</p> <pre>S3AsyncClient s3 =     S3AsyncClient.builder().build();  tmBuilder.s3Client(s3);</pre>
Selalu hitung multipart md5	<pre>tmBuilder.withAlwaysCalculateMultipartMd5(...); tmBuilder.setAlwaysCalculateMultipartMd5(...);</pre>	Tidak didukung.

## Perubahan perilaku







Transfer paralel membutuhkan klien AWS S3 berbasis CRT

[Dalam SDK for Java 2.x, fitur transfer paralel otomatis \(multipart upload/download\) tersedia melalui klien S3 berbasis CRT.AWS](#) Untuk mengaktifkan fitur transfer paralel, Anda harus secara eksplisit menambahkan dependensi [pustaka AWS Common Runtime \(CRT\)](#) untuk kinerja yang dimaksimalkan.

Klien S3 AWS berbasis CRT saja — tanpa menggunakan — memberikan `S3TransferManager` kinerja transfer paralel yang maksimal. `S3TransferManagerV2` menyediakan API tambahan yang memudahkan untuk mentransfer file dan direktori.

Kemampuan `S3TransferManager` untuk melakukan transfer paralel tergantung pada bagaimana `S3TransferManager` dimulai dan jika pustaka AWS Common Runtime (CRT) telah dinyatakan sebagai dependensi.

Tabel berikut menjelaskan tiga skenario inisialisasi untuk `S3TransferManager` v2 dengan dan tanpa AWS CRT dinyatakan sebagai dependensi.

Pendekatan inisialisasi S3 TransferManager v2	Apakah AWS CRT dinyatakan sebagai ketergantungan?	
	ya	tidak
<p>Inisialisasi <b>S3TransferManager</b> tanpa melewati instance <b>S3AsyncClient</b></p> <p>Metode membuat statis:</p> <pre>S3TransferManager.create();</pre> <p>- ATAU -</p> <p>Metode pembangun:</p> <pre>S3TransferManager.builder().build();</pre>	 <p>transfer paralel otomatis diaktifkan</p>	 <p>transfer paralel otomatis dinonaktifkan</p>
<p>Lewati <b>S3AsyncClient</b> instance yang dibangun dengan salah satu metode pembuat crt* ()</p> <pre>S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder().build(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre> <p>- ATAU -</p> <pre>S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre>	 <p>transfer paralel otomatis diaktifkan</p>	 <p>kesalahan runtime</p>
<p>Lulus <b>S3AsyncClient</b> instance yang dibangun dengan salah satu metode pembangun standar sehingga manajer transfer tidak memiliki referensi ke CRT</p>		

Pendekatan inialisasi S3 TransferManager v2	Apakah AWS CRT dinyatakan sebagai ketergantungan?	
<pre data-bbox="121 262 1015 451">S3AsyncClient s3AsyncClient = S3AsyncClient.builder().build(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre> <p data-bbox="121 493 243 535">- ATAU -</p> <pre data-bbox="121 567 1015 766">S3AsyncClient s3AsyncClient = S3AsyncClient.create(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre>	transfer paralel otomatis dinonaktifkan	transfer paralel otomatis dinonaktifkan

## Unduhan paralel melalui pengambilan rentang byte

Ketika fitur transfer paralel otomatis diaktifkan, S3 Transfer Manager v2 menggunakan [byte-range fetches](#) untuk mengambil bagian tertentu dari objek secara paralel (unduh multipart). Cara objek diunduh dengan v2 tidak tergantung pada bagaimana objek awalnya diunggah. Semua unduhan dapat memperoleh manfaat dari throughput dan konkurensi yang tinggi.

Sebaliknya, dengan S3 Transfer Manager v1, penting bagaimana objek awalnya diunggah. S3 Transfer Manager v1 mengambil bagian-bagian objek dengan cara yang sama seperti bagian-bagian yang diunggah. Jika objek awalnya diunggah sebagai objek tunggal, S3 Transfer Manager v1 tidak dapat mempercepat proses pengunduhan dengan menggunakan sub-permintaan.

## Perilaku kegagalan

Dengan S3 Transfer Manager v1, permintaan transfer direktori gagal jika ada sub-permintaan yang gagal. Tidak seperti v1, future return from S3 Transfer Manager v2 berhasil diselesaikan meskipun beberapa sub-permintaan gagal.

Akibatnya, Anda harus memeriksa kesalahan dalam respons dengan menggunakan [CompletedDirectoryDownload.failedTransfers\(\)](#) metode atau [CompletedDirectoryUpload.failedTransfers\(\)](#) metode bahkan ketika future selesai dengan sukses.

## Perubahan utilitas metadata EC2 dari versi 1 ke versi 2

Topik ini merinci perubahan utilitas metadata SDK for Java Amazon Elastic Compute Cloud (EC2) dari versi 1 (v1) ke versi 2 (v2).

### Perubahan tingkat tinggi

Perubahan	v1	v2
Ketergantungan Maven	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;       &lt;version&gt; 1.12.587<sup>1</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;     &lt;artifact Id&gt;aws-java-sdk-co re&lt;/artifactId&gt;   &lt;/dependency&gt; &lt;/dependencies&gt; </pre>	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt;       &lt;version&gt; 2.21.21<sup>2</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;imds&lt;/artifactId&gt;   &lt;/dependency&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;     &lt;artifact Id&gt;apache-client<sup>3</sup>&lt;/ artifactId&gt; </pre>

Perubahan	v1	v2
		<pre>&lt;/dependency&gt; &lt;/dependencies&gt;</pre>
Nama paket	com.amazonaws.util	software.amazon.awssdk.imds
Pendekatan instantiasi	<p>Gunakan metode utilitas statis; tidak ada instantiasi:</p> <pre>String localHostName =     EC2Metada     taUtils.getLocalHo     stName();</pre>	<p>Gunakan metode pabrik statis:</p> <pre>Ec2MetadataClient     client = Ec2Metada     taClient.create();</pre> <p>Atau gunakan pendekatan pembangun:</p> <pre>Ec2MetadataClient     client = Ec2Metada     taClient.builder()         .endpointMode(Endp     ointMode.IPV6)         .build();</pre>
Jenis klien	Hanya metode utilitas sinkron: EC2MetadataUtils	<p>Sinkron: Ec2MetadataClient</p> <p>Asinkron: Ec2MetadataAsyncClient</p>

<sup>1</sup> [Versi terbaru](#). <sup>2</sup> [Versi terbaru](#).

<sup>3</sup> Perhatikan deklarasi `apache-client` modul untuk v2. V2 dari utilitas metadata EC2 memerlukan implementasi `SdkHttpClient` antarmuka untuk klien metadata sinkron, atau antarmuka untuk klien metadata `SdkAsyncHttpClient` asinkron. [???](#)Bagian ini menunjukkan daftar klien HTTP yang dapat Anda gunakan.



## Meminta metadata

Di v1, Anda menggunakan metode statis yang tidak menerima parameter untuk meminta metadata untuk sumber daya EC2. Sebaliknya, Anda perlu menentukan jalur ke sumber daya EC2 sebagai parameter di v2. Tabel berikut menunjukkan pendekatan yang berbeda.

v1	v2
<pre>String userMetaData = EC2MetadataUtils.getUserData();</pre>	<pre>Ec2MetadataClient client = Ec2MetadataClient.create(); Ec2MetadataResponse response =     client.get("/latest/user-data"); String userMetaData =     response.asString();</pre>

Lihat [kategori metadata instance](#) untuk menemukan jalur yang perlu Anda berikan untuk meminta sepotong metadata.

### Note

Ketika Anda menggunakan klien metadata instance di v2, Anda harus bertujuan untuk menggunakan klien yang sama untuk semua permintaan untuk mengambil metadata.

## Perubahan perilaku

### Data JSON

Pada EC2, Layanan Metadata Instance (IMDS) yang berjalan secara lokal mengembalikan beberapa metadata sebagai string berformat JSON. Salah satu contohnya adalah metadata dinamis dari dokumen [identitas instance](#).

API v1 berisi metode terpisah untuk setiap bagian metadata identitas instance, sedangkan API v2 secara langsung mengembalikan string JSON. Untuk bekerja dengan string JSON, Anda dapat menggunakan [Document API](#) untuk mengurai respons dan menavigasi struktur JSON.

Tabel berikut membandingkan bagaimana Anda mengambil metadata dari dokumen identitas instance di v1 dan v2.

Kasus penggunaan	v1	v2
Ambil Wilayah	<pre>InstanceInfo instanceInfo =     EC2MetadataUtils.getInstanceInfo(); String region =     instanceInfo.getRegion();</pre>	<pre>Ec2MetadataResponse response =     client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String region =     instanceInfo.asMap().get("region").asString();</pre>
Ambil id instance	<pre>InstanceInfo instanceInfo =     EC2MetadataUtils.getInstanceInfo(); String instanceId =     instanceInfo.getInstanceId();</pre>	<pre>Ec2MetadataResponse response =     client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String instanceId =     instanceInfo.asMap().get("instanceId").asString();</pre>
Ambil jenis instance	<pre>InstanceInfo instanceInfo =     EC2MetadataUtils.getInstanceInfo(); String instanceType =     instanceInfo.getInstanceType();</pre>	<pre>Ec2MetadataResponse response =     client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String instanceType =     instanceInfo.asMap</pre>

Kasus penggunaan	v1	v2
		<code>() .get("instanceType").asString();</code>

### Perbedaan resolusi titik akhir

Tabel berikut menunjukkan lokasi yang diperiksa SDK untuk menyelesaikan titik akhir IMDS. Lokasi tercantum dalam prioritas menurun.

v1	v2
Properti sistem: <code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	Metode konfigurasi pembangun klien: <code>endpoint(...)</code>
Variabel lingkungan: <code>AWS_EC2_METADATA_SERVICE_ENDPOINT</code>	Properti sistem: <code>aws.ec2MetadataServiceEndpoint</code>
Nilai Default: <code>http://169.254.169.254</code>	File konfigurasi: <code>~.aws/config</code> dengan pengaturan <code>ec2_metadata_service_endpoint</code>
	Nilai yang terkait dengan diselesaikan <code>endpoint-mode</code>
	Nilai default: <code>http://169.254.169.254</code>

### Resolusi titik akhir di v2

Saat Anda secara eksplisit menetapkan titik akhir dengan menggunakan pembuat, nilai titik akhir tersebut diprioritaskan di atas semua pengaturan lainnya. Ketika kode berikut dijalankan, properti `aws.ec2MetadataServiceEndpoint` sistem dan `ec2_metadata_service_endpoint` pengaturan file konfigurasi diabaikan jika ada.

```
Ec2MetadataClient client = Ec2MetadataClient
    .builder()
    .endpoint(URI.create("endpoint.to.use"))
```

```
.build();
```

## Modus titik akhir

Dengan v2, Anda dapat menentukan mode titik akhir untuk mengonfigurasi klien metadata untuk menggunakan nilai titik akhir default untuk IPv4 atau IPv6. Mode titik akhir tidak tersedia untuk v1. Nilai default yang digunakan untuk IPv4 adalah `http://169.254.169.254` dan `http://[fd00:ec2::254]` untuk IPv6.

Tabel berikut menunjukkan berbagai cara yang dapat Anda atur mode endpoint dalam urutan prioritas menurun.

		Kemungkinan nilai
Metode konfigurasi pembangun klien: <code>endpointMode(...)</code>	<pre>Ec2MetadataClient client = Ec2MetadataClient .builder() .endpointMode(EndpointMode.IPV4) .build();</pre>	<code>EndpointMode.IPV4</code> , <code>EndpointMode.IPV6</code>
Properti sistem	<code>aws.ec2MetadataServiceEndpointMode</code>	IPv4, IPv6 (kasus tidak masalah)
Berkas Config: <code>~/.aws/config</code>	Setelan <code>ec2_metadata_service_endpoint</code>	IPv4, IPv6 (kasus tidak masalah)
Tidak ditentukan dengan cara sebelumnya	IPv4 digunakan	

## Bagaimana SDK menyelesaikan **endpoint** atau di v2 **endpoint-mode**

1. SDK menggunakan nilai yang Anda tetapkan dalam kode pada pembuat klien dan mengabaikan pengaturan eksternal apa pun. Karena SDK melempar pengecualian jika keduanya `endpoint` dan `endpointMode` dipanggil pada pembuat klien, SDK menggunakan nilai titik akhir dari metode mana pun yang Anda gunakan.

2. Jika Anda tidak menetapkan nilai dalam kode, SDK melihat ke konfigurasi eksternal—pertama untuk properti sistem dan kemudian untuk pengaturan dalam file konfigurasi.
  - a. SDK pertama-tama memeriksa nilai titik akhir. Jika nilai ditemukan, itu digunakan.
  - b. Jika SDK masih belum menemukan nilai, SDK akan mencari pengaturan mode titik akhir.
3. Terakhir, jika SDK tidak menemukan pengaturan eksternal dan Anda belum mengonfigurasi klien metadata dalam kode, SDK menggunakan nilai IPv4 dari `http://169.254.169.254`

## IMDSv2

Amazon EC2 mendefinisikan dua pendekatan untuk mengakses metadata instans:

- Layanan Metadata Instans Versi 1 (IMDSv1) - pendekatan permintaan/respons
- Layanan Metadata Instance Versi 2 (IMDSv2) — Pendekatan berorientasi sesi

Tabel berikut membandingkan bagaimana SDK Java bekerja dengan IMDS.

v1	v2
IMDSv2 digunakan secara default	Selalu menggunakan IMDSv2
Mencoba mengambil token sesi untuk setiap permintaan dan kembali ke IMDSv1 jika gagal mengambil token sesi	Menyimpan token sesi dalam cache internal yang digunakan kembali untuk beberapa permintaan

SDK for Java 2.x hanya mendukung IMDSv2 dan tidak kembali ke IMDSv1.

## Perbedaan konfigurasi

Tabel berikut mencantumkan opsi konfigurasi yang berbeda.

Konfigurasi	v1	v2
Percobaan ulang	Konfigurasi tidak tersedia	Dapat dikonfigurasi melalui metode pembangun <code>retryPolicy(...)</code>

Konfigurasi	v1	v2
HTTP	Batas waktu koneksi dapat dikonfigurasi melalui variabel <code>AWS_METADATA_SERVICE_TIMEOUT</code> lingkungan. Defaultnya adalah 1 detik.	Konfigurasi tersedia dengan meneruskan klien HTTP ke metode <code>builderHttpClient(...)</code> . Batas waktu koneksi default untuk klien HTTP adalah 2 detik.

## Contoh konfigurasi HTTP v2

Contoh berikut menunjukkan bagaimana Anda dapat mengkonfigurasi klien metadata. Contoh ini mengkonfigurasi batas waktu koneksi dan menggunakan klien HTTP Apache.

```

SdkHttpClient httpClient = ApacheHttpClient.builder()
    .connectionTimeout(Duration.ofSeconds(1))
    .build();

Ec2MetadataClient imdsClient = Ec2MetadataClient.builder()
    .httpClient(httpClient)
    .build();

```

## Perubahan di Amazon CloudFront presigning dari versi 1 ke versi 2

Topik ini merinci perubahan di Amazon CloudFront dari versi 1 (v1) ke versi 2 (v2).

### Perubahan tingkat tinggi

Perubahan	v1	v2
Ketergantungan Maven	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt; </pre>	<pre> &lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt; </pre>

Perubahan	v1	v2
	<pre> &lt;version&gt; 1.12.587<sup>1</sup>&lt;/version&gt; &lt;type&gt;pom&lt;/ type&gt; &lt;scope&gt;im port&lt;/scope&gt; &lt;/dependency&gt; &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt; &lt;dependency&gt; &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt; &lt;artifact Id&gt;cloudfront&lt;/art ifactId&gt; &lt;/dependency&gt; &lt;/dependencies&gt; </pre>	<pre> &lt;version&gt; 2.21.21<sup>2</sup>&lt;/version&gt; &lt;type&gt;pom&lt;/ type&gt; &lt;scope&gt;im port&lt;/scope&gt; &lt;/dependency&gt; &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt; &lt;dependency&gt; &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt; &lt;artifact Id&gt;cloudfront&lt;/art ifactId&gt; &lt;/dependency&gt; &lt;/dependencies&gt; </pre>
Nama paket	com.amazonaws.serv ices.cloudfront	software.amazon.aw ssdk.services.clou dfront
Nama kelas	<a href="#">CloudFrontUrlSigner</a> <a href="#">CloudFrontCookieSigner</a>	<a href="#">CloudFrontUtilities</a> <a href="#">SignedUrl</a> <a href="#">CannedSignerRequest</a> <a href="#">CustomSignerRequest</a>

<sup>1</sup> [Versi terbaru.](#) <sup>2</sup> [Versi terbaru.](#)

## Perubahan API

Perilaku	v1	v2
Bangun permintaan kalengan	Argumen diteruskan langsung ke API.	<pre data-bbox="1073 338 1507 1052"> CannedSignerRequest cannedRequest =     CannedSig nerRequest.builder()      .resourceUrl(resou rceUrl)      .privateKey(privat eKey)      .keyPairId(keyPairId)      .expirationDate(ex pirationDate)      .build(); </pre>
Membangun permintaan kustom	Argumen diteruskan langsung ke API.	<pre data-bbox="1073 1094 1507 1829"> CustomSignerRequest customRequest =     CustomSig nerRequest.builder()      .resourceUrl(resou rceUrl)      .privateKey(keyFile)      .keyPairId(keyPairId)      .expirationDate(ex pirationDate)      .activeDate(active Date)      .ipRange(ipRange) </pre>



Perilaku	v1	v2
		<code>.build();</code>
Hasilkan URL yang ditandatangani (kalengan)	<pre>String signedUrl =     CloudFrontUrlSigner.getSignedURLWith     CannedPolicy(         resourceUrl,         keyPairId, privateKey,         expirationDate);</pre>	<pre>CloudFrontUtilities     cloudFrontUtilities =         CloudFrontUtilities         s.create();  SignedUrl signedUrl =      cloudFrontUtilities     s.getSignedUrlWith     CannedPolicy(cannedRequest);  String url = signedUrl .url();</pre>
Hasilkan cookie yang ditandatangani (kustom)	<pre>CookiesForCustomPolicy     cookies =         CloudFrontCookieSigner.getCookiesFor         CustomPolicy(             resourceUrl,             privateKey, keyPairId             , expirationDate,             activeDate,             ipRange);</pre>	<pre>CloudFrontUtilities     cloudFrontUtilities =         CloudFrontUtilities         s.create();  CookiesForCustomPolicy     cookies =         cloudFrontUtilities     s.getCookiesForCustomPolicy(customRequest);</pre>

Header cookie yang difaktorkan ulang di v2

Di Java v1, Java SDK memberikan header cookie sebagai `file. Map.Entry<String, String>`

```
Map.Entry<String, String> signatureMap = cookies.getSignature();
String signatureKey = signatureMap.getKey(); // "CloudFront-Signature"
String signatureValue = signatureMap.getValue(); // "[SIGNATURE_VALUE]"
```

## Java v2 SDK memberikan seluruh header sebagai satu. String

```
String signatureHeaderValue = cookies.signatureHeaderValue(); // "CloudFront-
Signature=[SIGNATURE_VALUE]"
```

## Perubahan dalam mengurai URI Amazon S3 dari versi 1 ke versi 2

Topik ini merinci perubahan dalam mengurai URI Amazon S3 dari versi 1 (v1) ke versi 2 (v2.).

### Perubahan tingkat tinggi

Untuk mulai mengurai URI S3 di v1, Anda membuat instance AmazonS3URI dengan menggunakan konstruktor. Di v2 Anda memanggil `parseUri()` instance `S3Utilities`, untuk mengembalikan `fileS3URI`.

Perubahan	v1	v2
Ketergantungan Maven	<pre>&lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;       &lt;artifact Id&gt;aws-java-sdk-bom&lt;/ artifactId&gt;       &lt;version&gt; 1.12.587<sup>1</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; com.amazonaws&lt;/gro upId&gt;</pre>	<pre>&lt;dependencyManagement&gt;   &lt;dependencies&gt;     &lt;dependency&gt;       &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;       &lt;artifact Id&gt;bom&lt;/artifactId&gt;       &lt;version&gt; 2.21.21<sup>2</sup>&lt;/version&gt;       &lt;type&gt;pom&lt;/ type&gt;       &lt;scope&gt;im port&lt;/scope&gt;     &lt;/dependency&gt;   &lt;/dependencies&gt; &lt;/dependencyManageme nt&gt; &lt;dependencies&gt;   &lt;dependency&gt;     &lt;groupId&gt; software.amazon.aw ssdk&lt;/groupId&gt;</pre>

Perubahan	v1	v2
	<pre>&lt;artifact Id&gt;s3&lt;/artifactId&gt; &lt;/dependency&gt; &lt;/dependencies&gt;</pre>	<pre>&lt;artifact Id&gt;s3&lt;/artifactId&gt; &lt;/dependency&gt; &lt;/dependencies&gt;</pre>
Nama paket	com.amazonaws.serv ices.s3	software.amazon.aw ssdk.services.s3
Nama kelas	<a href="#">AmazonS3URI</a>	<a href="#">S3URI</a>

<sup>1</sup> [Versi terbaru.](#) <sup>2</sup> [Versi terbaru.](#)

## Perubahan API

Perilaku	v1	v2
Mengurai URI S3.	<pre>URI uri = URI.creat e( "https://s3.amazon aws.com");  AmazonS3Uri s3Uri =     new AmazonS3U RI(uri, false);</pre>	<pre>S3Client s3Client =     S3Client.create(); S3Utilities s3Utiliti es =     s3Client.utilities ();  S3Uri s3Uri =     s3Utilities.parseU ri(uri);</pre>
Ambil nama bucket dari URI S3.	<pre>String bucket = s3Uri.getBucket();</pre>	<pre>Optional&lt;String&gt; bucket = s3Uri.bucket();</pre>
Ambil kuncinya.	<pre>String key = s3Uri.get Key();</pre>	<pre>Optional&lt;String&gt; key = s3Uri.key();</pre>
Ambil kembali wilayah tersebut.	<pre>String region = s3Uri.getRegion();</pre>	<pre>Optional&lt;Region&gt; region = s3Uri.region();</pre>

Perilaku	v1	v2
		<pre>String region; if (s3Uri.region().isPresent()) {     region = s3Uri.region().get().id(); }</pre>
Ambil kembali apakah URI S3 adalah gaya jalur.	<pre>boolean isPathStyle = s3Uri.isPathStyle();</pre>	<pre>boolean isPathStyle = s3Uri.isPathStyle();</pre>
Ambil ID versi.	<pre>String versionId = s3Uri.getVersionId();</pre>	<pre>Optional&lt;String&gt; versionId = s3Uri.firstMatchingRawQueryParameter( "versionId");</pre>
Ambil parameter kueri.	N/A	<pre>Map&lt;String, List&lt;String&gt;&gt; queryParams = s3Uri.rawQueryParameters();</pre>

## Perubahan perilaku

### Pengkodean URL

v1 menyediakan opsi untuk meneruskan bendera untuk menentukan apakah URI harus dikodekan URL. Nilai default-nya adalah `true`.

Di v2, pengkodean URL tidak didukung. Jika Anda bekerja dengan kunci objek atau parameter kueri yang memiliki karakter cadangan atau tidak aman, Anda harus menyandikannya dengan URL. Misalnya Anda perlu mengganti spasi putih " " dengan. `%20`

## Gunakan SDK for Java 1.x and 2.x side-by-side

Anda dapat menggunakan kedua versi AWS SDK for Java dalam proyek Anda.

Berikut ini menunjukkan contoh `pom.xml` file untuk proyek yang menggunakan Amazon S3 dari versi 1.x dan DynamoDB dari versi 2.16.1.

### Example Contoh POM

Contoh ini menunjukkan entri `pom.xml` file untuk proyek yang menggunakan SDK versi 1.x dan 2.x.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId>
      <version>1.12.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.16.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-s3</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>dynamodb</artifactId>
  </dependency>
</dependencies>
```

# Kunci OpenPGP untuk AWS SDK for Java

Semua artefak Maven yang tersedia untuk umum ditandatangani menggunakan standar AWS SDK for Java OpenPGP. Kunci publik yang Anda perlukan untuk memverifikasi tanda tangan artefak tersedia di bagian berikut.

## Kunci saat ini

Tabel berikut menunjukkan informasi kunci OpenPGP untuk rilis SDK for Java 1x saat ini dan SDK for Java 2.x.

ID Kunci	0xac107b386692dadd
Tipe	RSA
Ukuran	4096/4096
Dibuat	30/06/2016
Kedaluwarsa	2024-10-08
ID Pengguna	AWSSDK dan Alat < aws-dr-tools@amazon .com>
Sidik jari kunci	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 DADD

Untuk menyalin kunci publik OpenPGP berikut untuk SDK for Java ke clipboard, pilih ikon “Salin” di sudut kanan atas.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
xsFNBFd1gAUBEACqbmmFbxkJgz1lD7wrlskQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJlMYp0viSWsX2psgvdmeyUpW9ap01rThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKR0mR0DzUuaokLPo24pfm9bnr1RnRrtwt5ktPAA5bM9ZZaGKriej
kT2lPffBbjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
u6PewUe2WP1nxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9b0ie
```

```
nj3QxLuQ1ZUKF79ES6JaM4t0z1gGcQeU1+Uk1gjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+ff+Xf0Cl6by0JFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFEMAuzzd0QJ4uz4xFFvaS0SZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVB0HKrq+fluorWepQivctzN6Y1N0kx5naTPGGaKWK7G2q
TbcY5SMnkIwFLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zsFNBFd1gAUBEAC8zNARpWb3dPMThL2xAY+fs60vXdb1Sk0tYJpDWPfgvo0d+VQ+
hV6Xu1GAHAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go
7xHIxgFjC046/06Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FK
VYR/j9uenEC/2NBcLuFy3q6cDfmCoDE0062kXMnaGz3knzEK/X1SkcjsxRDq7zaQ
lQ1Kou+3dICwy4x5SJQ8j1+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjy
pUwgp0MTo25gWxkvJlSJKU0b6b1786WnySIzF2gxqlkkEmB14RAssQkeXjrSmGws
MDyHNqyJeYFus18sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6Qxa
Zje9YSZUijGmZIdEBleRVt3Svhi8MY1nasd4bW2RK1sr7plkBf8QRe6biiQRf3KD
0Sn5CbmXpAchJ1ZHzRRdkXZDNQC6vCJxsy1300TrhJtAV1Yq347uyUbVi291ISVg
roUVtprismHoEk5Go0THbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbtnbs
/Hd981FdVghYYvq//gTakJk0WxfGq030wtXRndPOA0T+qhP3TE+LtGRJ+wARAQAB
wsF1BBgBCgAPBQJXdYAFaHsMBQkHhh+AAAoJEKwQezhmktrdTyEP/0H0VHwQsaW
jMrGj00MFzXGUo8SBmYYTBs29VM8wBGDsPkYCjeZzU16i9iqDpDqxyqmTigcjH
V8CDx/6xsMBLG2yKaKZ4m3+Yn0Qf/sQkyCvqiyMF9mS7pDYWy+mPhPuw8TDIfiqg
VhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+NAM6Q5dYkCebyvwzLmg1sVni
16iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNViJ9zAaPI78X9v6PpDGn0kg6oL
zrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJB1kke6kw9+KagY8mrVX1ZenRg
+sY/4vxJreYWQeq167ggx+wFjKDcfhZA7m70LH0DysrGVCLcmuinUBaN1HmLDcGY
XZ+kMCoXf0bpuCVByQmNJgEb47EIFlx/+TEeNHKM0+22xL1atFzXfkEVZck+NghL
ZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7GNpuiEFUYh69QQ2//CS5H51o
sC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02NK0fvF/IKHnGkvH28rv00PCv0
WTA/MClv28y0PrSvcmXnduLtkBEX7TISMPW+n+0Ta63/z4YFFEZ7sFLrEm3Q3vJ
MN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=Z9u3
-----END PGP PUBLIC KEY BLOCK-----
```

## Riwayat dokumen

Topik ini menjelaskan perubahan penting pada Panduan AWS SDK for Java Pengembang selama sejarahnya.

Panduan ini terakhir diterbitkan pada 4 Maret 2024.

Perubahan	Deskripsi	Tanggal
<a href="#">the section called “Aman”</a>	Tambahkan instruksi untuk menonaktifkan IMDSv1.	Maret 14, 2024
<a href="#">the section called “tep-by-step Instruksi S”</a>	Tambahkan instruksi step-by-step migrasi.	8 Maret 2024
<a href="#">Migrasi ke versi 2</a>	Perbarui topik migrasi.	Februari, 14, 2024
<a href="#">the section called “Konfigurasi AWS klien HTTP berbasis CRT”</a>	Tambahkan informasi tentang klien HTTP AWS berbasis CRT sinkron.	Januari 5, 2024
<a href="#">the section called “Identitas Amazon Cognito” dan the section called “Penyedia Identitas Amazon Cognito”</a>	Contoh Amazon Cognito dipindahkan ke bagian Contoh Kode.	28 Desember 2023
<a href="#">Gunakan fitur SDK</a>	Mengolah ulang topik fitur SDK.	Desember 11, 2023
<a href="#">Kunci OpenPGP</a>	Berikan kunci OpenPGP saat ini.	6 Desember 2023
<a href="#">the section called “Perubahan serialisasi”</a>	Jelaskan perbedaan serialisasi antara v1 dan v2 SDK for Java.	5 Desember 2023
<a href="#">the section called “Manajer Transfer S3”</a>	Tambahkan bagian yang merinci perubahan di S3	13 November 2023



Perubahan	Deskripsi	Tanggal
	Transfer Manager dari versi 1 ke versi 2.	
<a href="#">the section called “Referensi anotasi”</a>	Tambahkan daftar anotasi kelas data yang dapat digunakan dengan DynamoDB Enhanced Client.	30 Oktober 2023
<a href="#">???</a>	Tambahkan informasi tentang status migrasi pustaka dan utilitas dari SDK for Java v1.x ke v2.x	17 Oktober 2023
<a href="#">???</a>	Memperbarui topik penyiapan Gradle	17 Oktober 2023
<a href="#">the section called “Abaikan atribut null dari objek bersarang”</a>	Tambahkan informasi tentang anotasi DynamoDB Enhanced Client. <code>@DynamoDbIgnoreNulls</code>	September 22, 2023
<a href="#">the section called “Akses Lintas Wilayah”</a>	Tambahkan informasi tentang akses lintas wilayah ke bucket Amazon S3.	31 Agustus 2023
<a href="#">the section called “Pertahankan benda kosong”</a>	Tambahkan bagian yang membahas <code>@DynamoDbPreserveEmptyObject</code> anotasi.	Agustus 25, 2023
<a href="#">???</a>	Perbarui bagian klien layanan.	15 Agustus 2023

Perubahan	Deskripsi	Tanggal
<a href="#">the section called “Rekomendasi klien”</a>	Sejak versi 0.23, AWS CRT mendukung OS berbasis musl seperti Alpine Linux. Rekomendasi klien HTTP sekarang mencerminkan dukungan musl.	11 Agustus 2023
<a href="#">the section called “Buat kebijakan IAM”</a>	Tambahkan bagian API Pembuat Kebijakan IAM	31 Juli 2023
<a href="#">the section called “Memulai”</a>	Perbaiki beberapa cuplikan di bagian Memulai topik DynamoDB Enhanced Client.	Juli 24, 2023
<a href="#">the section called “Dukungan proxy”</a>	Tambahkan informasi dukungan proxy HTTP dan contoh untuk setiap klien HTTP.	Juni 2, 2023
Atur ulang daftar isi	Promosikan <a href="#">Contoh kode</a> bagian dan <a href="#">Bekerja dengan Layanan AWS</a> ke entri TOC tingkat atas.	24 Mei 2023
<a href="#">the section called “Tambahkan ketergantungan logging”</a>	Tampilkan dependensi Gradle di bagian logging.	Mei 23, 2023
<a href="#">the section called “Bekerja dengan hasil paginasi”</a>	Perbarui topik pagination.	18 Mei 2023
<a href="#">the section called “Siapkan proyek Gradle”</a>	Perbarui penyiapan proyek Gradle.	3 Mei 2023
<a href="#">DynamoDB API Klien yang Ditingkatkan</a>	Topik DynamoDB Enhanced Client API yang ditulis ulang dirilis.	28 April 2023

Perubahan	Deskripsi	Tanggal
<a href="#">Perbarui instruksi tutorial mulai</a>	Pola dasar Maven dimodifikasi untuk menyertakan opsi untuk CredentialsProvider; instruksi dimodifikasi sesuai.	11 April 2023
<a href="#">the section called “Rekomendasi klien”</a>	Tambahkan panduan keputusan klien HTTP	30 Maret 2023
Pembaruan praktik terbaik IAM	Panduan yang diperbarui untuk menyelaraskan dengan praktik terbaik IAM. Untuk informasi selengkapnya, lihat <a href="#">Praktik terbaik keamanan di IAM</a> .	14 Maret 2023
<a href="#">the section called “Muat ulang kredensi profil”</a>	Tambahkan bagian tentang memuat ulang kredensial profil.	9 Februari 2023
<a href="#">the section called “Konfigurasi AWS klien HTTP berbasis CRT”</a>	Perbarui topik untuk rilis GA.	Februari 8, 2023
<a href="#">the section called “Bekerja dengan metadata instans Amazon EC2”</a>	Tambahkan contoh terpandu untuk klien Java SDK untuk layanan metadata instans Amazon S3.	1 Februari 2023
<a href="#">the section called “Gunakan klien S3 yang berkinerja”</a>	Tambahkan bagian untuk Klien AWS S3 berbasis CRT.	19 Desember 2022
<a href="#">the section called “Transfer file dan direktori”</a>	Perbarui contoh Manajer Transfer Amazon S3 untuk rilis GA.	19 Desember 2022
<a href="#">the section called “Praktik terbaik”</a>	Ditambahkan bagian praktik terbaik.	18 November 2022

Perubahan	Deskripsi	Tanggal
<a href="#">the section called “Memuat kredensi sementara dari proses eksternal”</a>	Menambahkan bagian tentang pemuatan kredensial dari proses eksternal.	15 November 2022
<a href="#">the section called “Metrik klien layanan”</a>	Daftar metrik yang diperbarui dengan persyaratan penggunaan klien HTTP.	9 November 2022
<a href="#">the section called “Transfer file dan direktori”</a>	Contoh kode dikoreksi.	2 November 2022
<a href="#">the section called “Kurangi waktu startup SDK untuk AWS Lambda”</a>	Bagian yang diperbarui dengan opsi tambahan untuk mengurangi waktu startup Lambda.	1 November 2022
<a href="#">the section called “Klien HTTP”</a>	Menambahkan informasi konfigurasi untuk mencakup semua klien HTTP di SDK.	26 Oktober 2022
<a href="#">the section called “Pencatatan log”</a>	Topik logging diperbarui untuk menyertakan rincian wire logging untuk semua klien HTTP.	4 Oktober 2022
<a href="#">the section called “AWSlayanan basis data”</a>	Ditambahkan bagian ikhtisar layanan AWS database dan SDK for Java 2.x.	13 September 2022
<a href="#">EC2-Classic Networking Akan Pensiun</a>	EC2-Classic akan pensiun pada 15 Agustus 2022.	28 Juli 2022
<a href="#">the section called “Opsi otentikasi tambahan”</a>	Pembaruan ke ketergantungan yang diperlukan untuk otentikasi masuk tunggal.	18 Juli 2022

Perubahan	Deskripsi	Tanggal
<a href="#">the section called “Keamanan Lapisan Pengangkutan (TLS)”</a>	Perbarui informasi keamanan TLS.	8 April 2022
<a href="#">the section called “Opsi otentikasi tambahan”</a>	Menambahkan informasi lebih lanjut tentang pengaturan dan penggunaan kredensial.	22 Februari 2021
<a href="#">the section called “Menyiapkan proyek GraAlvm Native Image”</a>	Topik baru untuk menyiapkan proyek Gambar Asli GraalVM.	18 Februari 2021
<a href="#">the section called “Polling untuk negara sumber daya”</a>	Pelayan dirilis; menambahkan topik untuk fitur baru.	30 September 2020
<a href="#">the section called “Gunakan metrik SDK”</a>	Metrik dirilis; menambahkan topik untuk fitur baru.	17 Agustus 2020
<a href="#">the section called “Amazon SNS”</a>	Ditambahkan contoh topik untuk Amazon SNS.	Mei 30, 2020
<a href="#">the section called “Kurangi waktu startup SDK untuk AWS Lambda”</a>	Ditambahkan topik kinerja AWS Lambda fungsi.	29 Mei 2020
<a href="#">the section called “Mengatur JVM TTL untuk pencarian nama DNS”</a>	Menambahkan topik caching DNS JVM TTL.	27 April 2020
<a href="#">the section called “Siapkan proyek Apache Maven”, the section called “Siapkan proyek Gradle”</a>	Maven dan Gradle baru menyiapkan topik.	21 April 2020
<a href="#">the section called “Keamanan Lapisan Pengangkutan (TLS)”</a>	Menambahkan TLS 1.2 ke bagian keamanan.	19 Maret 2020

Perubahan	Deskripsi	Tanggal
<a href="#">the section called “Berlangganan Amazon Kinesis Data Streams”</a>	Menambahkan contoh Kinesis aliran.	2 Agustus 2018
<a href="#">the section called “Bekerja dengan hasil paginasi”</a>	Ditambahkan auto pagination topik.	5 April 2018
<a href="#">???</a>	Ditambahkan contoh topik untuk IAM, Amazon EC2, CloudWatch dan DynamoDB.	29 Desember 2017
<a href="#">the section called “Amazon S3”</a>	Ditambahkan getObject contoh untuk Amazon S3.	Agustus 7, 2017
<a href="#">the section called “Gunakan pemrograman asinkron”</a>	Ditambahkan topik async.	Parquet
Rilis GA dari <a href="#">AWS SDK for Java 2.x</a>	AWS SDK for Java versi 2 (v2) dirilis.	28 Juni 2017

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.